

Run-time Allocation of Buffer Resources for Maximizing Video Clip Quality in a Wireless Last-hop System

Clark N. Taylor and Sujit Dey
Electrical and Computer Engineering
University of California, San Diego

{cntaylor, dey}@ece.ucsd.edu

Abstract—The transmission of digital video over wireless channels faces many problems common to both wired and wireless networks such as bandwidth and buffer restrictions and low energy constraints, together with problems unique to wireless technologies such as extreme and rapidly varying channel noise. In this paper, we focus on enabling the wireless communication of video clips despite the noisy and dynamic nature of wireless channels. Many efforts have been made in the past to increase the error resiliency of compressed video, but these approaches have not considered the effects of buffering on the received video quality. Traditional buffering (where a “sliding window” of the next few frames is kept in the buffer at all times), while useful for overcoming latency jitter, cannot be used to guarantee against poor video quality and stalls due to wireless channel fluctuations. To help overcome the weaknesses of traditional buffering, we introduced *Out-of-oRder Buffering (ORBit)*, a method for selecting important frames from throughout a video clip and buffering them before playback begins. ORBit can lead to significant gains in video quality despite variations in the wireless channel during the communication of a video clip, but requires the judicious selection of ORBit frames, with their corresponding quantization and channel coding levels. To enable run-time selection of ORBit frames, we introduce a fast video distortion estimation technique. Using our estimation technique, we present an ORBit decision algorithm that selects ORBit frames depending on the wireless technology and end-user appliance constraints. Using the ORBit decision algorithm, significant gains of up to 9dB in video quality are obtained.

I. INTRODUCTION

Due to the recent exponential growth in the number of mobile phone subscribers, fixed line subscribers were outnumbered by mobile subscribers for the first time during the year 2002[1]. Simultaneously, the Internet continues to grow in popularity and usefulness as more and more content becomes available on the Internet. Therefore, it is widely expected that wireless access to the Internet, including the many rich multimedia resources available on the Internet, will soon become common and ubiquitous.

On the other hand, the transmission of multimedia data, especially streaming video, over wireless channels is a challenging and unsolved problem. Wireless channels typically have low bandwidth and a large and rapidly varying amount of noise, leading to significant problems when transmitting video. In this paper, we focus on enabling the communication of short digital video clips (up to 5 minutes) over the Internet with a wireless last-hop channel. We focus on video clips because of the large possible future market for wireless video clips (news clips, location-aware bulletins, multimedia message services, etc). In addition, video clips provide more opportunities for optimization than the more general streaming video case.

Current research addressing the confluence of wireless channels and digital video have focused primarily on one of two areas. First, many research works attempt to address the low-bandwidth requirements of wireless channels, resulting in low-bitrate digital video standards such as MPEG-4 [2] and H.263[3]. Second, much research has been performed on minimizing the effect of wireless channel errors during the transmission of digital video [4], [5], [6], [7]. Both of these research areas have led to significant improvements in the quality of video received over a wireless channel. In this paper, however, we attempt to further improve the video quality received by a wireless appliance through the judicious allocation of buffering resources in the appliance. All former methods of improving the video quality cited here can also be used in conjunction with our method.

Previous work on the buffering of digital video has primarily focused on ensuring that overflow (too much information for resource-constrained appliances) and underflow (the emptying of the buffer when more information is needed) do not occur during the transmission of variable-bit rate video [8], [9]. However, the previous buffering research does not consider other types of buffering besides traditional buffering.

In this paper, we investigate the tradeoff between traditional buffering (where the first few frames of a video clip are stored before video display begins), and a new video buffering scheme, *Out-of-oRder Buffering (ORBit)*[10]. ORBit increases the video quality by transmitting frames selected from the entire video clip at the beginning of the video clip, causing the selected frames to be displayed at a higher quality independent of wireless errors or packet dropping occurring later in the video clip. However, the method for selecting ORBit frames introduced in

[10] was ad-hoc. In this paper, we introduce a systematic algorithm that attempts to maximize the received video quality by judiciously selecting what frames should be ORBit frames, and associating a quantization and channel coding level with each frame. Any remaining buffer resources are allocated towards traditional buffering.

The paper is organized as follows. Section II reviews the two types of buffering: (1) traditional and (2) ORBit, and their effects on video quality. In section III, we discuss a quick and efficient method for estimating the received quality of video when transmitted over the Internet and a wireless channel. Section IV presents our algorithm that balances traditional and ORBit buffering by selecting ORBit frames, with their quantization and channel coding levels, that maximize the received video quality. Results are presented demonstrating the improved video quality achieved through judicious buffer allocation. Section V concludes the paper.

II. THE BUFFERING OF VIDEO CLIPS AND ITS EFFECTS

When a user clicks on a hyper-link to watch a video, he/she usually has to wait for a few seconds before the video display begins. During this wait time, buffering occurs. In this section, we look at two possible ways that buffering can be performed, traditional and ORBit, and analyze the effects of each method.

A. Traditional Buffering

Traditional buffering consists of the storage of compressed video data in a first-in, first-out (FIFO) manner before the data is to be decoded, creating a “sliding window” of buffered data. For example, if 5 seconds of traditional buffering occurs before video decoding begins, then the traditional buffering FIFO (TBF) will have 5 seconds of compressed video data in it. As video decoding begins, information will begin to be removed from the TBF, while at the same time new information is received by the appliance and placed into the TBF. Therefore, when communication and decoding conditions are in perfect equilibrium, the TBF will maintain a balance of 5 seconds throughout. Because different frames in a video clip are compressed to different sizes, however, the video decoder may sometimes pull out data faster than data arrives, or vice versa. In general, however, we assume that the average rates of input and output to the TBF are matched to create an equilibrium.

To determine what the effects of traditional buffering on video quality are, we simulated different amounts of traditional buffering, with different latency jitter parameters. The simulator breaks up a video clip into packets, and assigns each packet a different amount of time taken for transmission according to a left-truncated Gaussian distribution, with a mean of 1s and left-truncation point of 0.1s. To represent different amounts of latency jitter, different standard deviations are chosen for the Gaussian distribution. The video decoder is started a certain time period after the beginning of the simulator, and decodes a frame every 1/30th of second. The decoder discards any packets which arrive after their corresponding video frame has already been decoded.

Fig. 1 represents the results of these simulations, with the x axis representing the time elapsed before the video decoder begins (in seconds), the y axis showing the average video quality in terms of PSNR, and each plot representing a different latency standard deviation parameter (a larger standard deviation leads to greater latency jitter). Each data point is averaged across 5 different runs of the simulator. It is important to note that for each latency jitter plot, as the available buffer size increases, the video quality also increases until an “elbow” point is reached. Once this elbow point is reached, an increase in the amount of traditional buffering does not correspond with an increase in video quality.

B. Out-of-oRder Buffering (ORBit)

During wireless video clip communication, the conditions (such as bandwidth allowed, bit error rate, etc) can vary dramatically over time, requiring more robust ways of communicating digital video. To allow for more robust communication of video clips, ORBit, a method for improving the received video quality of a wirelessly transmitted video clip was introduced[10]. To implement ORBit, important frames from throughout a video clip are transmitted at the beginning of the video clip as shown in Fig. 2. All ORBit frames must be transmitted before traditional buffering and transmission of the video clip can begin.

This work supported by the SRC, Task # 900.001

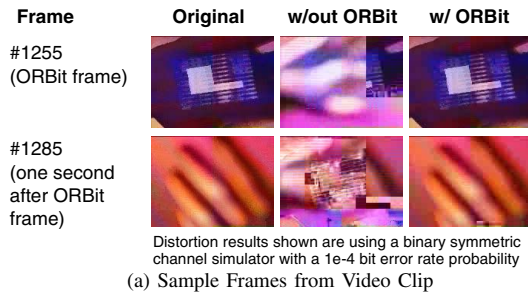


Fig. 3. Effect of ORBit Method on Video Quality

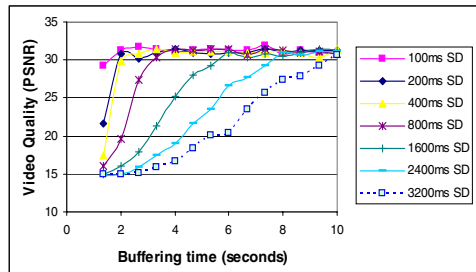


Fig. 1. Effect of traditional buffer size on video quality with differing amounts of jitter

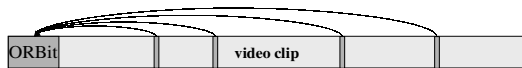


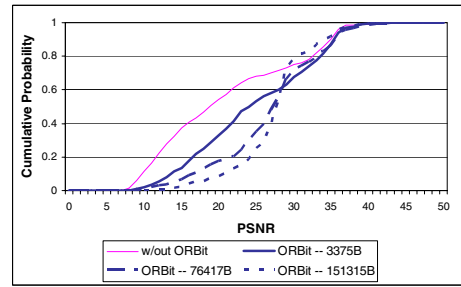
Fig. 2. The Out-of-oRder Buffering (ORBit) technique

For each frame selected as an ORBit frame, a marker denoting its exact temporal location in the video clip is transmitted together with the frame. The frame is also intra-encoded so that it does not depend on any other information to be decoded and displayed by the video decoder. (In MPEG-4, this can be implemented using a Group of Pictures (GOP) header to denote the temporal location of the frame, and compressing the frame as an I-frame.) Therefore, the ORBit information buffered at the beginning of a video clip contains all the information about a frame, together with its temporal location in the video stream, allowing the decoder to recreate the ORBit frames independent of any information transmitted later on in the video clip.

To implement the display of ORBit frames during video playback, the ORBit frames are stored, while still compressed, in temporal order of display. When the decoder of a video clip decodes a frame at the same temporal location as an ORBit frame, the quality of the decoded frame is compared with the buffered ORBit frame, and the higher quality of the two is displayed to the end-user. In addition, the higher quality frame is used as a reference when decompressing inter-coded frames later in the video clip. To determine which frame is higher quality, the decoder first checks to see if there have been any errors which might affect the currently decoded frame. If errors have occurred then the ORBit frame information is used. If there were no errors affecting the current decoded frame, then the quantization levels used to compress the two frames are compared. The frame with the lower quantization level is then used for both display and as the reference frame for further inter-frame decoding.

Figure 3 demonstrates the effect of ORBit. In Fig. 3(a), we show two sample frames from a video clip communicated over a wireless channel with a $1E-4$ bit error rate. The three columns represent the original frame, the decoded frame when ORBit was not used (traditional buffering only), and the decoded frame when the ORBit method was used. The first row represents a frame which was selected as an ORBit frame. The second row represents a frame located a second after the ORBit frame in the first row. As can be seen, the addition of ORBit frames can result in significant improvements in video quality for both the frames selected as ORBit frames and the frames following the ORBit frames.

For a more detailed idea of the effects of ORBit, we simulated the transmission of a 150kbps, 68.5s video clip over a wireless channel modeled by a 2-state Markov model oscillating between a bit error rate of $1E-6$ and $1E-3$. In Fig. 3(b), we show the cumulative probability distribution (cdf) graph for a video frame's quality in terms of PSNR, with each data point being averaged across 10 simulation runs. The thin plot represents the cdf for a video transmitted without any ORBit frames, while the thick plots represent the cdf's for transmissions using ORBit



(b) Cumulative Probability of Frame's Video Quality

frames, with 3375, 76417, and 151315 bytes allocated during buffering to ORBit frames. For all four plots, enough traditional buffering was used to overcome all latency jitter effects. As shown, the addition of ORBit frames can significantly decrease the probability of low-quality frames being viewed by the end-user. It is important to note, however, that ORBit does not increase the PSNR for frames which already have a high PSNR. ORBit can therefore be viewed as a safeguard against low-quality frames.

While the ORBit method can significantly increase the received video quality, it comes at a cost. To transmit ORBit frames requires buffering to occur before the beginning of the video clip, thereby either lengthening the user's latency before the video clip begins displaying, or reducing the amount of buffer space that can be allocated for traditional buffering. Therefore, it is necessary to develop an algorithm that judiciously allocates available buffer space between ORBit and traditional buffering to maximize the received video quality. To make decisions regarding the insertion of ORBit frames, it is necessary to estimate the received video quality achieved when transmitting a video clip. We present our fast runtime method for estimating video quality in the next section.

III. FADE: A FAST ESTIMATION OF THE RECEIVED VIDEO QUALITY OF A TRANSMITTED VIDEO CLIP

The selection of ORBit frames and the allocation of time for traditional buffering is a process that occurs at the video server, before the actual transmission of the video clip. Because the video clip is not yet transmitted, the effect of adding ORBit frames and traditional buffering on the received video quality must be estimated. In this section, we introduce an analytical method for estimating the received video quality of a video clip. This involves three main components, each described in detail below: (1) Frame-based Distortion Estimation (FaDE), (2) the effect of traditional buffering, and (3) the effect of ORBit frames.

A. Frame-based Distortion Estimation (FaDE)

The Frame-based Distortion Estimation technique (FaDE) introduced here attempts to estimate the mean square error (MSE), a common metric for measuring image and video quality, for each individual frame in the video clip. To estimate the MSE of a given frame, the probability of errors occurring during the transmission of that frame must be known.

We consider two possible sources of errors that may occur during the transmission of a video frame. First, we assume that the RTP and UDP protocols[11] are used for the transmission of packets. When packets are dropped in transit across the Internet, UDP will not ask for a retransmission, allowing for dropped packets, leading to a possible source of errors.

Second, the wireless channel may, due to noise, corrupt some of the packets in transit. This can lead to two separate effects. First, the UDP layer may, upon performing a checksum of a corrupted packet, drop the packet. Second, if the UDP checksum is not performed, and the packet is passed onto the video decoder, then when an error is detected, the video decoder will drop either the whole packet or some portion of the packet. Therefore, errors occurring during the transmission of packets across the wireless channel can also be modeled as dropped packets.

While using TCP for the transmission of video would eliminate packet errors, TCP demonstrates several weaknesses for video transmission. When a packet is dropped or corrupted, TCP will try to have the packet retransmitted, leading to reduced bandwidth, higher latency jitter, and time periods where no video is received as TCP tries to retransmit lost or corrupted packets (manifested to the end-user as a video which begins playing, then suddenly pauses for several seconds while the buffer fills up again before it begins playing). These problems are avoided by RTP and UDP, but at the cost of errors being introduced to the video stream seen by the end-user.

By combining the probabilities of error due to packet dropping and corruption, an application-layer packet error rate representing the communication effects of a wireless last-hop system is obtained (P). Once the probability of a packet error is known for a given video frame (i), the received MSE (Rec_MSE_i) can be estimated using the following equation

$$Rec_MSE_i = (1 - P) \cdot Trans_MSE_i + P \cdot f(Rec_MSE_{i-1}, \dots)$$

if the frame was intra-coded (basically a compressed image), or

$$Rec_MSE_i = (1 - P) \cdot g(Trans_MSE_i, Rec_MSE_{i-1}) + P \cdot f(Rec_MSE_{i-1}, \dots)$$

if the frame is encoded as a P-frame. In both formulas, $Trans_MSE_i$ is the MSE of frame i after encoding, before any errors have occurred, f is a function which determines the MSE of a received frame when an error has occurred and g determines what the MSE of the current frame is depending on the past frame's MSE (because P-frames are inter-coded) and the current encoded data. Note that these formulas are recursive as Rec_MSE_i depends on Rec_MSE_{i-1} .

In our estimation model, we define g as simply $g() = Rec_MSE_{i-1}$. In other words, we assume that a P-frame, when correctly received, will essentially maintain the same MSE that was present in the previous frame. The function f , on the other hand is more difficult. First off, the function f depends on what type of error concealment is in use by the video decoder. Therefore, the function f must determine, given Rec_MSE_{i-1} and the error concealment method, what the Rec_MSE_i would be if all packets were dropped.

For this work, we assume that if a packet is received incorrectly, the corresponding macroblocks are concealed by copying the same blocks from the previous frame. We also define a quantity, Mean Square Difference (MSD_i), which contains the MSE between the original (before encoding) i and $i - 1$ frames, representing the error introduced through error concealment.

To determine the MSE of an error-concealed frame ($f(Rec_MSE_{i-1}, MSD_i)$), we first model Rec_MSE_{i-1} and MSD_i as the mean squared error of a random variable. The random variable is defined as a Gaussian distribution, centered at each possible pixel value, and truncated at 0 and 255 (the possible range of pixel values). Therefore, the expressions defining Rec_MSE_{i-1} and MSD_i are:

$$Rec_MSE_{i-1} = \frac{1}{255} \int_0^{255} \int_0^{255} (x - z)^2 \frac{G(\frac{x-z}{\sigma_x})}{\Phi(\frac{-z}{\sigma_x}) - \Phi(\frac{255-z}{\sigma_x})} dx dz$$

$$MSD_i = \frac{1}{255} \int_0^{255} \int_0^{255} (y - z)^2 \frac{G(\frac{y-z}{\sigma_y})}{\Phi(\frac{-z}{\sigma_y}) - \Phi(\frac{255-z}{\sigma_y})} dy dz$$

where G is the probability density function of a normalized Gaussian distribution, and Φ is the Gaussian cumulative distribution function. In these formulas, Z is a random variable representing the original (unencoded) pixels in frame $i - 1$, Y represents the original pixels in frame i , and X represents the pixels received for frame $i - 1$. Another way to look at this is that Z is a uniform distribution over all pixel values (0 to 255), and therefore both Rec_MSE_{i-1} and MSD_i can be expressed solely in terms of their Gaussian standard deviation (σ_x and σ_y , respectively). After the random variables X and Y have been defined by their associated σ 's, Rec_MSE_i can be computed by finding the MSE between random variables Y and X , yielding

$$f(Rec_MSE_{i-1}, MSD_i) = \frac{1}{255} \int_0^{255} \int_0^{255} \int_0^{255} (y - x)^2 \cdot \frac{G(\frac{x-z}{\sigma_x})}{\Phi(\frac{-z}{\sigma_x}) - \Phi(\frac{255-z}{\sigma_x})} \frac{G(\frac{y-z}{\sigma_y})}{\Phi(\frac{-z}{\sigma_y}) - \Phi(\frac{255-z}{\sigma_y})} dx dy dz$$

Unfortunately, this expression does not integrate symbolically, so we numerically integrated it for all combinations of MSE values between 0 and 10800, spaced apart by 25 (0,25,50,...10800) and stored the results in a lookup table. The table is approximately 187,000 entries, and through the use of interpolation, completely represents the function $f(Rec_MSE_{i-1}, MSD_i)$. This table is also general for all video clips and does not need to be regenerated from clip to clip.

To demonstrate the effectiveness of our technique, we simulated the transmission of a video clip with a 10% error rate. We compared the average MSE results across 10 simulation runs (Simulated) with our proposed estimation technique (FaDE) and a technique previously proposed in the literature (Rose[12]). The results are shown in Fig. 4

where the x-axis is the frame number in the video clip, and the y-axis is the MSE in dB (PSNR). Notice that both the Rose and FaDE techniques follow the actual simulated results. The Rose technique, however, is a pixel-based technique and therefore runs approximately 4000x slower than our proposed method (13 vs 0.003 seconds).

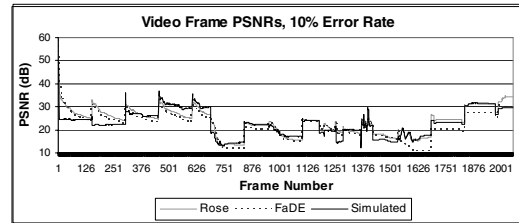


Fig. 4. Accuracy of our proposed Fast Distortion Estimation (FaDE) method compared against the Rose[12] technique and simulated results

Once the MSE for each individual frame has been computed, FaDE returns a *Weighted Average MSE* to represent the video quality of the entire video clip. Within a video clip, certain frames are more important to the "content" of the clip than other frames. Therefore, in bad wireless conditions, it is more important for these frames to be at high quality than some of the other frames. For example, during a news clip that shows a jet flying overhead, a frame that shows the jet in the sky is more important than the frames before and after the jet flies overhead. In the worst case, the video may degrade to a type of slide show, with the key content frames being more clear than the rest of the video. Therefore, the content provider can pass to FaDE a "weighting factor" for each frame that denotes the importance of content in each frame. This weighting factor for each frame is then used to create the Weighted Average MSE that FaDE returns. Similarly, we denote Weighted Peak Signal to Noise Ratio (WPSNR) as the PSNR computed when Weighted Average MSE is used as the noise figure.

B. Estimating the Effect of Traditional Buffering

To estimate the effect of traditional buffering on received video quality, we need to estimate a new packet drop rate dependent on the amount of traditional buffering used. Traditional buffering affects the relationship between when a packet arrives and when the packet is needed by the video decoder. Therefore, traditional buffering controls how many packets get dropped because they arrived too late. For example, if there is 3 seconds worth of traditional buffering, and a packet takes 2 seconds to arrive, then the packet will have to wait 1 second before being used by the video decoder. On the other hand, if traditional buffering is set for 1 second and a packet takes 2 seconds to arrive, then the packet will be dropped by the video decoder because it arrived too late.

Assuming that the latency distribution of packets traveling across the Internet and a wireless last-hop can be represented by a left-truncated Gaussian distribution, the probability of packet drops due to traditional buffering is

$$P_{TB} = \frac{1.0 - \Phi(\frac{TB - Lat_mean}{\sigma_{Lat}})}{1.0 - \Phi(\frac{Lat_LT - Lat_mean}{\sigma_{Lat}})}$$

where P_{TB} is the probability of a packet dropping due to not enough traditional buffering, TB is the time allocated to traditional buffering and Lat_mean is the mean, Lat_LT is the left truncation point, and σ_{Lat} is the standard deviation of the Gaussian distribution representing latency. If with an infinite amount of traditional buffering, there is a probability P of packets being dropped, then with finite traditional buffering, the total probability of packets dropping (P_{new}) is

$$P_{new} = (1 - P) \cdot P_{TB}$$

P_{new} is then used by the FaDE method proposed in section III-A to estimate the video quality of each frame in the video clip.

C. Estimating the Effect of ORBit Frames

To estimate the effect of ORBit frames being added, the FaDE method proposed in section III-A is modified in four ways. First, the MSE computation for the frame chosen as an ORBit frame is modified to compute the distortion for an I-frame rather than a P-frame. Second, $Trans_MSE_i$ is replaced by the MSE of the ORBit frame when transmitted. Third, the size of the ORBit frames, at the chosen quantization and channel coding levels must be estimated, converted to time (by dividing by the video clip's bit rate), and subtracted from the traditional buffering time allocated. P_{new} must be then be recomputed and used by FaDE. Last, if channel coding is used to further protect the ORBit frame, a probability

of error for that particular frame must be determined and used by the FaDE method.

For our current implementation, we assume that a Reed Solomon RS(15,t) channel codec is used for the ORBit frames, where t is a parameter chosen by the algorithm introduced in section IV. RS(15) codes have two useful properties that helped us decide to use RS(15). First, because in RS(15) each codeword is 4 bits, the codeword evenly divides up each byte, thereby avoiding any bit alignment complications. Second, because we need to channel code across packets to overcome packet erasures (due to dropped packets), not within packets to overcome bit errors, a smaller code (as compared to RS(255), which also has the desired bit alignment properties) was easier to use when determining error resiliency and space overhead.

To determine the probability of packet errors when using channel coding, we must determine what the probability of more than 2t packet erasures occurring is, and what the packet error rate is for those cases. The expression for determining the new packet error probability (P_{CC_err}) is:

$$P_{CC_err} = \sum_{i=2t+1}^{15} \frac{i}{15} \frac{15!}{(15-i)!i!} P_{new}^i (1 - P_{new})^{15-i},$$

where P_{CC_err} is the probability of error for that ORBit frame with channel coding, P_{new} is the packet error rate (dependent on the size of traditional buffering) without channel coding, and t is the error correction strength of the Reed-Solomon code. P_{CC_err} is then passed to the generic FaDE method to compute the distortion reduction caused by adding an ORBit frame, with a specific channel coding. The bit overhead due to channel coding was computed by multiplying the size of a frame time $15/(15-2t)$

Using the methods proposed in this section, it is possible to estimate the distortion of any combination of ORBit frames and traditional buffering. In the following section, we present an efficient algorithm that maximizes the video quality achieved through the usage of ORBit and traditional buffering.

IV. AN EFFICIENT ORBIT DECISION ALGORITHM FOR MAXIMIZING VIDEO QUALITY

In this section, we present a run-time ORBit decision algorithm used for maximizing the video quality of video clips transmitted over the Internet with a wireless last-hop. The ORBit decision algorithm has the following inputs: (1) the buffer space allowed for the current communication by the end-user appliance, (2) the wireless technology being used and its worst-case channel conditions, and (3) the current latency conditions (represented as latency standard deviation and mean parameters). Using these inputs, the algorithm generates a list of ORBit frames, with their associated quantization and channel coding levels, that maximizes video quality.

In addition to the formal specification of the algorithm given above, the ORBit decision algorithm must also be extremely fast at run-time. Therefore, the ORBit decision algorithm is divided into two portions: (1) the precomputation stage, and (2) the run-time stage. These two stages are presented below, followed by the results demonstrating the effectiveness of this optimization algorithm.

A. Precomputation Stage of ORBit Decision Algorithm

The goal of the precomputation stage of the ORBit decision algorithm is to generate an ordered list of ORBit frames for a given wireless technology. The ordered list that is generated is used by the run-time portion of the algorithm to quickly decide on which ORBit frames, at what quantization and channel coding levels, should be transmitted to maximize video quality. Because the precomputation stage cannot know what the wireless channel conditions will be during transmission of a video clip, the precomputation stage assumes the packet error rate due to the worst-case wireless channel conditions. As discussed in section II-B, ORBit is useful primarily as a safeguard against bad wireless conditions, so the selection of ORBit frames is designed to guarantee video quality even during very bad channel conditions. Therefore, the precomputation stage is run once for each possible wireless technology, at the worst packet error rate for that technology.

The precomputation stage, shown in Fig. 5 is implemented as a greedy algorithm. For every frame in the video clip, the improvement in distortion if that frame is chosen as an ORBit frame is estimated using FaDE and divided by the size required to transmit the frame. This quantity represents the gain obtained by adding that frame at a given quantization and channel coding level. For each frame, the gain is computed assuming the current quantization level and the next lowest quantization level if that frame was previously selected as an ORBit frame, or simply the maximum quantization level if the frame has not already been selected

as an ORBit frame. (Note that higher quantization levels mean less space and lower video quality than lower quantization levels, so the algorithm works from the worst frame quality toward better frame quality.) For each frame being considered, the distortion is estimated for each channel coding level (denoted by t as explained in section III-C) starting at 0 and working up to 7, or until the gain is no longer improving as the channel coding level increases. After checking every frame, for the quantization and channel coding levels as explained, the frame with the largest gain is chosen. If the gain is positive (gain>0) and the maximum number of frames has not been exceeded (num_frames<max), then the frame, quantization level, and channel coding level with the largest gain are added to the ordered list (the ‘‘Store Frame, Quant, and CC to List’’ step in Fig. 5) and the search repeats. Otherwise the search terminates. Calls made to FaDE after adding ORBit frames to the ordered list assume that all previous frames in the ordered list are used as ORBit frames by FaDE, representing the gain in video quality already achieved by previous ORBit frame additions. During the precomputation stage, an infinite amount of traditional buffering is assumed, leading to a constant packet error rate independent of how many ORBit frames are added to the list during this stage.

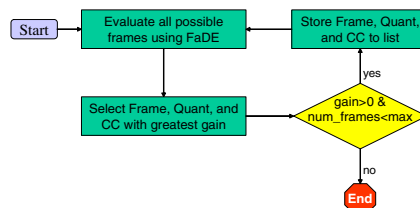


Fig. 5. Precomputation stage of ORBit decision algorithm

Note that while the precomputation stage is very computationally expensive, it is performed before run-time, and will therefore not affect the latency observed by a user during transmission of a video clip. Next, we present the run-time algorithm used to quickly select ORBit frames at run-time.

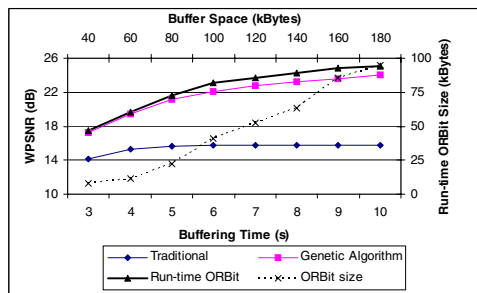
B. Run-time Stage of ORBit Decision Algorithm

At run-time, the ORBit decision algorithm is given the worst-case packet error rate (determined by the wireless technology being used), the latency conditions (estimated from network monitoring), and the available buffering space (given by the end-user appliance). Depending on the worst-case packet error rate, the run-time stage picks the corresponding ordered list generated by the precomputation stage. To determine how many ORBit frames from the ordered list to actually transmit, the run-time stage first assumes that all available buffer space is allocated for traditional buffering, and evaluates the video quality using FaDE. Starting at the beginning of the ordered list of ORBit frames, the run-time stage calculates the effect of adding each frame, and its corresponding decrease in traditional buffering size. As more ORBit frames are added, eventually the increase in packet error rate due to the decrease in traditional buffering will outweigh the advantages of adding ORBit frames. This point is manifested by FaDE returning an increase in distortion rather than a decrease when evaluating the ORBit frame. By searching through the ordered list of ORBit frames until the distortion starts increasing, the run-time stage selects which ORBit frames from the ordered list to transmit for the current video transmission. Note that while the addition of ORBit frames increases the buffering time required, it does not increase the bitrate requirements during transmission of the video clip.

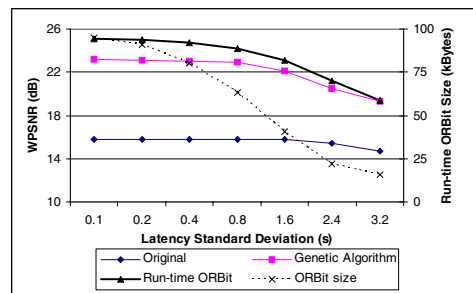
C. Results of ORBit Decision Algorithm

In this subsection, we present results demonstrating the improved video quality obtained and the low run-time cost of implementing our ORBit decision algorithm. The video clip used to obtain these results is an MPEG-4 compressed, 29.97 frames per second, 160 by 120 pixel, 68.5 second video stream containing several scene changes and high and low-motion portions. The video clip is also divided into 256 byte MPEG-4 packets to increase the error resiliency of the video clip. The weight array passed into FaDE for computing the Weighted Average MSE assumes each frame has a weight of 0.2, with 13 key content frames weighted at 1.0. For the frames following the key content frames, the weights decrease at a rate of 0.05 for each frame until it reaches 0.2 again, thereby ensuring a visible (longer than one frame) improvement in video quality. This gives the key content frames extra weight and forces the ORBit decision algorithm to try and maintain that frame quality for a longer period of time so that it is visible to the user.

The graphs in Fig. 6 show the video quality achieved through the use of the ORBit decision algorithm. The video quality results, in terms



(a) Latency SD = 1.6s, mean=1s, PER=20%



(b) Buffer Time=6s, Latency mean=1s, PER=20%

Fig. 6. Effect on video quality of ORBit decision algorithm compared to a genetic algorithm and traditional buffering only

of Weighted PSNR (y-axis), are compared for the traditional method of buffering only (Original), a genetic algorithm for choosing ORBit frames that used FaDE as its fitness function, and the run-time ORBit method. The amount of buffer space consumed by the ORBit frames when using our run-time ORBit method is also shown (ORBit size). A genetic algorithm was chosen for comparison because an exhaustive search for the optimal video quality would be too computationally expensive (over 2000! hours), so the genetic algorithm should give a good representation of what the optimal video quality results could be. In each of the plots, the MPEG-4 packet drop rate was 20% (representing a wireless bit error rate of $1E-4$), the latency mean was 1s, and the latency left-truncation point was 100ms. Note that while other methods such as TCP or ARQ could have been used to decrease the packet drop rate, this would lead to a significant reduction in bandwidth, starving the necessary bandwidth from the video decoder. In Fig. 6(a), the video quality achieved is shown for different total buffering times (x-axis) ranging from 3 to 10 seconds (40-180kBytes of storage), with a latency standard deviation of 1600ms. In Fig 6(b), a 6s (100kByte) buffer is assumed, with the latency standard deviation times being varied (x-axis). As shown in Fig. 6, the video quality improvement achieved through the ORBit decision algorithm is very high (up to 9 dB). In addition, the ORBit decision algorithm presented achieves video quality results even better than the genetic algorithm, which should be close to optimal, suggesting that the greedy approach outlined in sections IV-A and IV-B is not significantly sub-optimal. Finally, the buffer space consumed by the ORBit frames is varied effectively depending on the current latency conditions and the total buffer space available.

One possible source of sub-optimal video quality in the selection of ORBit frames is that worst-case channel conditions are always assumed. To evaluate the significance of this assumption, we simulated several scenarios where the realized bit error rate of the channel is lower than the worst-case error rate assumed when selecting ORBit frames. In Fig. 7, we show three plots of video quality in terms of weighted PSNR (y-axis), for different realized packet error rates (x-axis). For all data points, transmission properties of a 1s latency mean, 1.6s latency standard deviation, and 60kByte buffer size were assumed. The bottom plot, *Original Video Quality*, represents the video quality achieved if no ORBit frames are added. The top plot, *Selection with Foreknowledge*, assumes that the ORBit selection algorithm had perfect foreknowledge of the realized error conditions of the channel, and both the precomputation and run-time stages assume the realized error rate. The middle plot, *ORBit Selection Algorithm*, represents the video quality that results when the worst-case wireless channel conditions are assumed (PER=20%) during the precomputation and run-time stages of the ORBit selection algorithm. Note that even when the worst-case and realized conditions are widely divergent, significant video quality gains are still achieved while assuming worst-case conditions. For example, at a 2% realized packet error rate (a factor of 10x from the worst-case packet error rate), a gain of 3dB in video quality is achieved, compared to 4dB when the ORBit frames were selected with perfect foreknowledge. This demonstrates the ability of our ORBit selection algorithm to improve video quality even without a perfect foreknowledge of wireless conditions during the transmission of a video clip.

It is important to note that the ORBit selection algorithm is very efficient at run-time, taking 0.19 seconds to select the ORBit frames necessary to fill up 6 seconds of buffering time, under minimal latency jitter conditions (standard deviation = 0.1 seconds). As the latency jitter goes up, or the amount of buffer space goes down, the ORBit selection algorithm would take even less time to execute. By comparison, the genetic algorithm took more than 8 hours to select the frames for the current buffer conditions, and had still not fully converged (all run-times

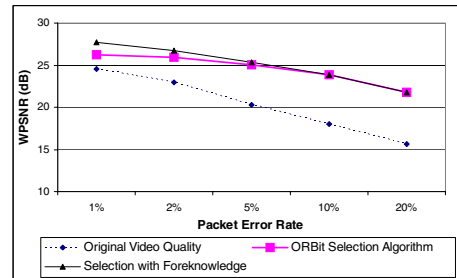


Fig. 7. Effect of assuming worst-case channel conditions when selecting ORBit frames

are for a Pentium IV, 2.8GHz, Redhat Linux 7.3 machine). The small run-time cost of our algorithm demonstrates the ability to select at run-time what ORBit frames should be transmitted.

As shown, the ORBit selection algorithm can lead to significantly improved video quality for several different buffer space and latency jitter conditions. In addition, significant gains in video quality are obtained even when the wireless channel conditions are not worst-case. Also, the run-time cost of ORBit frame selection was shown to be minimal, enabling selection of ORBit frames customized to the end-user appliance and network conditions.

V. CONCLUSION

In this paper, we have described an algorithm that maximizes the video quality of a video clip transmitted over the Internet with a wireless last-hop. To estimate the effects of different buffer allocations, a new Frame-based Distortion Estimation (FaDE) algorithm was introduced. Using FaDE, a greedy algorithm for selecting ORBit frames is proposed. Significant video quality improvements are achieved in a variety of communication conditions. The video quality results of our ORBit decision algorithm are comparable with a genetic algorithm based approach, while being several orders of magnitude faster.

REFERENCES

- [1] World Telecommunication Development Report: Reinventing Telecoms. http://www.itu.int/ITU-D/ict/publications/wtdr_02/, 2002. 6th edition.
- [2] The MPEG Home Page. <http://mpeg.telecomitalia.com/>.
- [3] ITU Telecom, Standardization Sector of ITU, "Video Coding for Low Bitrate Communication, Recommendation H.263 Version2", Feb 1998.
- [4] B. Girod and N. Färber, "Feedback-Based Error Control for Mobile Video Transmission", *Proceedings of the IEEE*, vol. 87, pp. 1707-23, October 1999.
- [5] Y. Wang and Q.-F. Zhu, "Error Control and Concealment for Video Communication: A Review", *Proceedings of the IEEE*, vol. 86, pp. 974-97, May 1998.
- [6] R. Talluri, "Error-Resilient Video Coding in the ISO MPEG-4 Standard", *IEEE Communications Magazine*, vol. 36, pp. 112-19, June 1998.
- [7] J. Brailean, "Wireless Multimedia Utilizing MPEG-4 Error Resilient Tools", in *Proceedings, 1999 Wireless Communications and Networking Conference*, pp. 104-8, September 1999.
- [8] W.-C. Feng, *Buffering Techniques for Delivery of Compressed Video in Video-on-Demand Systems*. Kluwer Academic Publishers, 1997.
- [9] S. Sen, et. al., "Online Smoothing of Variable-Bit-Rate Streaming Video", *Transactions on Multimedia*, vol. 2, pp. 37-48, March 2000.
- [10] C. N. Taylor and S. Dey, "ORBit: An Adaptive Method of Shaping Video Data for Transmission Over Imperfect Channels", in proceedings, 37th Asilomar Conference on Signals, Systems, and Computers, Nov. 2003.
- [11] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*. Boston: Addison-Wesley, 2000.
- [12] R. Zhang, S. L. Regunathan, and K. Rose, "Video Coding with Optimal Inter/Intra-Mode Switching for Packet Loss Resilience", *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 966-976, June 2000.