Expected Run-time Distortion Based Scheduling for Delivery of Scalable Media

Zhourong Miao and Antonio Ortega
Integrated Media Systems Center, Signal and Image Processing Institute,
Department of Electrical Engineering-Systems,
University of Southern California, Los Angeles, CA 90089-2564
{zmiao, ortega}@sipi.usc.edu

Abstract

Scalable, or layered, media representation is considered to be more suitable for transmission over heterogeneous networks. In this paper we study the problem of scalable layered streaming media delivery over a channel with packet loss or delay. The goal is to find a proper on-line packet scheduling policy for the server to transmit the packets such that the playback quality at the client is maximized. The problem requires taking into account the impact of the delay constrained delivery, data dependencies and channel conditions. We propose a framework for scalable streaming media delivery with a novel packet scheduling algorithm, referred as *Expected Run-time Distortion Based Scheduling* (ERDBS), which decides which packets should be sent (or retransmitted) at any given time. Our proposed algorithm summarizes the effect of the above constraints into a single metric, so that the packet scheduling decision can be made with very low computational complexity. Our approach improves playback quality about 2 dB as compared to a simple fixed scheduling approach. The proposed framework and the scheduling algorithm can be applied to both pre-recorded and real-time encoded streaming media applications.

Keywords – Streaming, scalable media, multimedia transmission, Internet

I. Introduction

Streaming media applications are becoming increasingly popular on the Internet. Streaming media data usually has the following properties: (i) the client can play back the media before it is fully downloaded, with a fixed start-up delay. Obviously, all the data packets have to arrive at the client end before their playback time. Therefore, this delay constraint requires sufficient channel bandwidth to deliver the streaming data on time. (ii) Use of retransmission to recover lost/corrupted packet is limited by the delay constraints. (iii) The media data packets may have different impacts on the quality of the reconstructed signal and therefore the loss of some packets may have a more severe impact on quality than the loss of others. To improve the performance of streaming applications, many research works developed various media coding schemes (including pre- and post-processing) and data delivery algorithms to reduce the playback quality degradation during periods of limited bandwidth or packet losses. For instance, scalable video (also called layered video), which is supported by the MPEG 4 standard [1], provides more flexibility for choosing the source data rate to be delivered over the channels with different available bandwidth.

In this paper we focus on the problem of delivery of scalable streaming media over a lossy packet network, e.g., the Internet. More specifically, we consider an end-to-end system, where a video server and client are connected through a channel, suffering from packet loss and varying channel bandwidth. Our goal is to find a packet transmission policy to select the packets to be transmitted (or retransmitted) at any given time during a streaming session, in such a way as to maximize the

This research has been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152 and by the National Science Foundation under grant MIP-9804959.

playback quality. In this paper we propose a server-driven "packet scheduling" algorithm especially designed for scalable media.

We introduce our main ideas with the help of several examples. First, consider a video sequence encoded in a single layer and without any temporal dependencies (e.g, motion JPEG). To transmit such a video object can be straightforward: we can simply transmit the frames according to their original their playback time, i.e., frames to be displayed earlier will be transmitted earlier (due to the delay constraint). A second example is to transmit video encoded with bi-directional temporal dependencies (e.g., MPEG encoded video). For example, when B-frames are used, it may be necessary to transmit all the reference frames before transmitting any of the B-frames. This is because B-frames are useless unless their reference frames have arrived.

The last example is for a scalable video encoded with both temporal and SNR (or other) dependencies, where each frame may contain several layers, e.g., the Fine Grain Scalability (FGS) algorithms with MPEG 4 [1]. The base layer can be decoded by itself with low quality; while the higher layer can be decoded, with the presence of the entire base layer, to obtain enhanced visual quality. The data dependencies among both layers and frames introduce a more complex dependency relationship across the different video data packets.

Furthermore, when streaming over a best-effort network with packet loss or error, retransmission can be used to recover the lost packets, as long as the delay constraints are met. However a retransmitted packet typically has an extra delay of one or more RTT(s) (round-trip-time), and can not be guaranteed to arrive the client on time. In addition, even if there is still time to retransmit a given packet, a decision needs to be made on whether this packet should be given more preference over other retransmission requests.

In this paper, we propose a scheduling algorithm to select the most "important" packets to be transmitted at a given time, i.e., the packets that will tend to maximize the playback quality, where "importance value" of a packet is evaluated by taking into account data dependencies, network conditions and other factors. The work in this paper is an extension of our previous work [2] where we apply the scheduling algorithm to layered video objects with more complex dependencies than those considered in [2]. Our simulation results show that by using the proposed algorithm, the video playback quality (in PSNR) can be improved around 2 dB, compared to a simple fixed scheduling approach, the when packet loss rate is around 20%.

Retransmission with delay constraints has been studied in [3], [4], where the decisions have been made on whether to retransmit or not based on the time-out factor the lost packet. While those works consider the problem of retransmissions, they do not consider specifically scalable media, which is addressed in our work.

In [5] a rate control algorithm for delivering MPEG-4 video over the Internet was proposed with a priority re-transmission strategy for recovery of lost packets, which considers the constraint to prevent the receiver buffer underflow. This is achieved by giving priority to retransmission of base (lower) layers. However this work did not address the problem of the delivery order of new packets in the sender's buffer. For example, the enhancement layer is packetized before transmission, and those packets actually have decoding dependencies on each other as well as the dependencies on the base layer. Our proposed framework solves the problem of delivery order of both the new and lost packets and can be the complement work for other rate-controlled based scheme which decides the rate of the video bit-stream to be transmitted, e.g., [5].

Optimization of layered streaming media delivery was also raised by Podolsky *et al* [6], who use a Markov chain analysis to find the optimal packets transmission and retransmission policies. However, the algorithm in [6] needs to search over all the possible candidate policies and thus the policy space grows exponentially with the number of layers and frames (in the scheduling window). In practice this algorithm may only be used with a limited number of layers and frames. Chou and Miao also addressed the same problem with a rate-distortion analysis [7], [8]. They show that the policy space can be factored so that packet dependencies are loosely coupled, and the optimal schedule policy can be found through a few of iterations over all the packets in the scheduling window. Therefore that framework can operate on more layers and a larger transmission window.

The main contribution of our work is to propose a simplified formulation and cost function so that a decision on which packet to transmit can be found without any iterations, reducing the complexity greatly with respect to [7], [8]. This is achieved by introducing the concept of expected run-time distortion, which summarizes the parameters to be considered in the packet scheduling (such as data dependencies, network condition) into a single packet importance metric. A similar concept of packet expected distortion has been proposed in [9], [10] and also used in [7], [8]. Those metrics take into account the status of packets whose decoding is needed before the current packet can be decoded (i.e., its parent packets). Our definition of expected run-time distortion extends this metric by explicitly including the "importance" of the children packets (i.e., those that depend on the current one). While the iterative techniques in [7], [8] resolve the data dependencies and take into account the impact of future scheduling of other packets, we can achieve these without iterations by using our proposed metric.

Therefore our proposed scheduling scheme can operate, with very low computational overhead, on more complex media streaming, such as MPEG 4 FGS scalable video, where a video packet may have hundreds of parent or children packets in the dependency tree (see details below). Furthermore, our proposed low complexity algorithm is no longer restricted to operate on fixed transmission times as in [6], [7], [8] (i.e., the scheduling algorithm assume packets sent at fixed intervals). Our packet selection algorithm can be used at any desired time when policy is needed to select packet(s) for transmission (or retransmission). Thus data packet size can be either fixed or variable.

The paper is organized as follows. In Section II we describe the backgrounds for scalable media (e.g., MPEG 4 FGS) data, the data packetization and the streaming system architecture used in this paper. Section III formulates our packet scheduling problem. Section IV presents the proposed ERDBS algorithm, and simulation results is shown in Section V. Section VI concludes the paper.

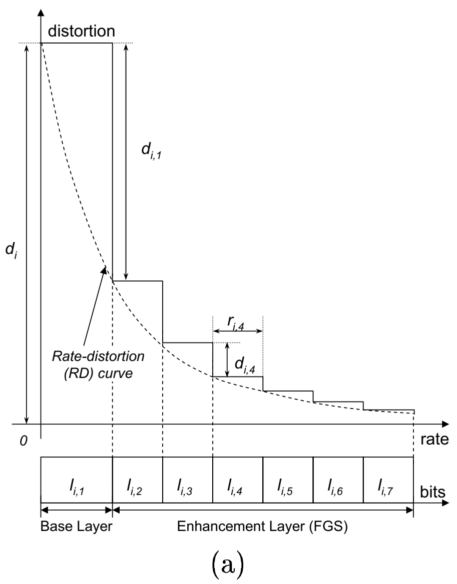
II. Scalable Media and Streaming System Architecture

Scalable encoded media has several layers. The lowest layer (or base layer) can be decoded by itself to obtain a coarse reconstructed version of the signal. As increasing numbers of higher layers to be decoded, more refined version of the signal is achieved. The MPEG 4 standard specifies a FGS (Fine Grain Scalability) mode [1], where a video sequence can be encoded into two layers: a base layer and an enhancement layer. The base layer has to be received completely before it can be decoded; while to obtain better reconstruction quality, one can continue to decode the enhancement layer, and can stop decoding at any position of the enhancement layer.

A. *Data packetization*

In this paper we use the MPEG 4 FGS video stream as implemented by the reference codec [1]. The base and enhancement layers are packetized separately, see Fig. 1(a). Usually the enhancement layer has large size and in order to generate different layers out of a single FGS layer, the enhancement layer is packetized into several packets; while the base layers of many frames can be packetized into a single packet.

For each packet, denoted as $l_{i,j}$ (the j^{th} layer of the i^{th} frame), we are interested in the following parameters: (i) the size of the packet $r_{i,j}$; (ii) the playback time of the packet t_i, which is defined as the time when frame i has to be displayed at the client end; (iii) the distortion value of the packet $d_{i,j}$; (iv) the set of its parent packets $\mathcal{A}_{i,j}$; and (v) the set of its children packets $\mathcal{B}_{i,j}$.



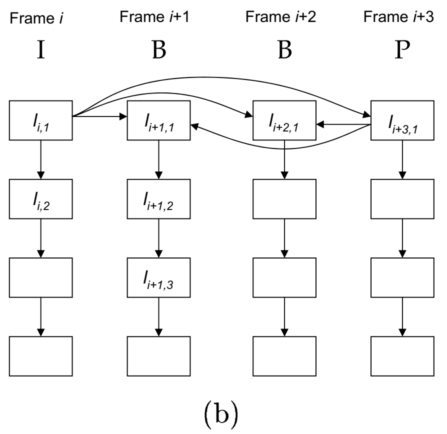


Fig. 1. (a) MPEG 4 FGS packetization of a single frame. (b) MPEG 4 FGS packetization of frames with inter-frame dependencies.

The packet distortion value $d_{i,j}$ is determined as the decrease in the distortion of the reconstructed signal when the corresponding packet $l_{i,j}$ is decoded. For example in Fig. 1(a), when packet $l_{i,4}$ is decoded, the distortion is reduced by $d_{i,4}$. If the base layer is split into several packets then the distortion value of each base layer packet can be assigned proportional to its packet size $r_{i,j}$, while their sum is equal to the total distortion value of the base layer. However it should be emphasized that this distortion value assignment is for our scheduling algorithm proposed below, decoding any single packet of the base layer cannot decrease the distortion of the reconstructed signal, as the base layer (of a frame) cannot be decoded without all its packets being available.

The parent set $\mathcal{A}_{i,j}$ of packet $l_{i,j}$ is defined as the set of packets that has to be available (or decoded) in order to decode packet $l_{i,j}$. For example $\mathcal{A}_{i,4}$, the parent set of packet $l_{i,4}$, includes packets $l_{i,1}$, $l_{i,2}$ and $l_{i,3}$. The child set $\mathcal{B}_{i,j}$ of packet $l_{i,j}$ is defined as the set of packets that cannot be decoded without the presence of packet $l_{i,j}$, i.e., $\mathcal{B}_{i,j}$ contains all packets that have packet $l_{i,j}$ in their parent set. For example, the child set of $l_{i,4}$ is $\mathcal{B}_{i,4}$ including packets $l_{i,5}$, $l_{i,6}$ and $l_{i,7}$.

Fig. 1(a) is an example for a single frame with only intra-frame dependencies (e.g., an I frame). For any P or B frame, the inter-frame dependencies should be taken into account to form the parent and child sets, which may contain packets across several frames. For example, in Fig. 1(b), when two packets are connected by a line, this means there is a direct dependency between them. The arrows are directed from parent to child packets. Note that some parent packets of a packet $l_{i,j}$ may not be shown explicitly, but should also be counted as well. For example, packet $l_{i+1,2}$ has a direct parent $l_{i+1,1}$ and other parent packets $l_{i,1}$ and $l_{i+3,1}$ (this is because packets $l_{i,1}$ and $l_{i+3,1}$ are in the parent set of packet $l_{i+1,1}$).

Although the original MPEG 4 FGS stream has two layers, after packetization, the enhancement layer can be broken into several sub-layers, where each packet represents a sub-layer. We define the total distortion of a frame, d_i, as the distortion when a frame is completely lost (see Fig. 1(a)). A frame reconstructed with all its layers (packets) will have the minimum distortion. Define the total distortion of the media stream as the sum of d_i of all the individual frames. The playback distortion is defined as the total distortion minus the distortion of all layers and frames that are actually used at playback. Define $a_{i,j}$ as an indicator such that $a_{i,j} = 1$ if $l_{i,j}$ is *used* for playback, otherwise $a_{i,j} = 0$. For a media sequence with N frames, with L_i layers in each frame, The *actual playback distortion*, D_V, can be obtained as

$$D_V = \sum_{i=1}^{N}\sum_{j=1}^{L_i} d_{i,j} - \sum_{i=1}^{N}\sum_{j=1}^{L_i} a_{i,j}d_{i,j}. \tag{1}$$

Note that the on-time arrival of a packet $l_{i,j}$ does not necessary mean that it can be used for playback, as decoding a packet is not possible unless all its parent packets have also arrived on time.

B. System architecture

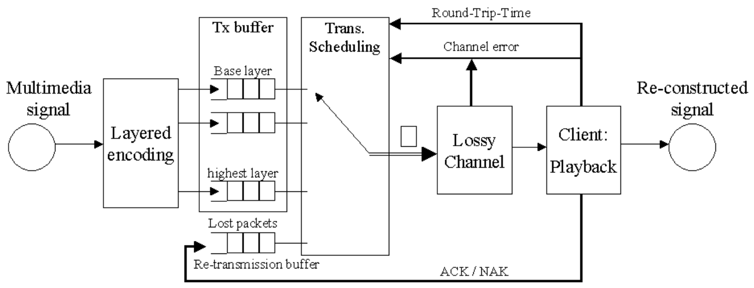


Fig. 2. System architecture

Fig. 2 shows the architecture of our proposed streaming system. The server and the client are connected by an unreliable channel link, which has low delay but with packet loss/delay (e.g., a UDP channel). We denote ϵ the packet loss probability in this server-client data channel. The source media sequence is compressed and packetized into several layers/packets, then is fed into the server's transmission buffer. These packets are referred as the "new" packets waiting to be scheduled for transmission. There are also packets in the server's buffer that are waiting for retransmission because they are reported lost or no acknowledgment was received. The server's scheduling module selects one packet at a transmission time from those buffers and sends it to the channel. At the client end, the lost or damaged packets are reported to the server via a feedback channel.

The value of round-trip-time (RTT) is defined as the interval from the time a packet is sent from the server to the time the server gets feedback on this packet from the client. With a smaller RTT, the server can get the feedback more promptly and have more time to re-send (if necessary) a packet before its time-out. If the feedback channel is also unreliable, RTT can be used by the sender as the "time-out" threshold in the case of lost of feedback. Both channel error and RTT can be estimated by the feedback information. They can be used by the server to adjust the stream delivery mechanism according to the varying channel conditions.

The start-up delay, τ, is defined as the period between the time that the first packet being sent by the server, and the time that the first frame starts to be displayed at the client. Usually a larger initial delay can smooth out more variations in channel behavior, and enable more time for retransmissions, resulting in a better playback quality. Refer to [11], [12], [13] for a more detailed analysis of the trade-off between start-up delay and playback quality. Our proposed scheduling framework can be applied for both pre-recorded and real-time encoded media which has a small start-up delay.

III. The Packet Scheduling Problem

The goal of this work is to minimize the playback distortion D_V for a streaming session in a lossy packet network. Due to the fixed start-up delay constraint, not all lost packets can be recovered by retransmission. However, if the server schedules a packet to be sent much earlier than its playback time, this packet will have more chances to be retransmitted (and received) before it is too late for display. This fact motivates the main ideas in our proposed framework.

Given a set of packets, $\mathcal{G}$, that are the candidates to be transmitted by the server, we define a schedule s as the transmission order of all those candidate packets, which specifies whether and when a packet should be transmitted. Clearly, the order of delivering the packets has an impact on the actual playback quality, because of the delay constraint (which limits the retransmission) and data dependencies. Intuitively, it would be useful to transmit (retransmit) a more "important" packet at an earlier time, e.g., a base layer or I frame packets, rather than simply transmit all the packets in a sequential order (i.e., following the original time stamp order) without any distinction among all the packets.

Due to the packet loss, The importance of a packet needs to take into account not only the playback distortion $d_{i,j}$ and its playback time t_i, but also the status of other packets in both the parent and child sets. These other parameters can be captured into a "expected" distortion metric. We denote $\hat{D}_G$ as the *expected playback distortion* of the packet set $\mathcal{G}$, where $\mathcal{G}$ is the set of packets currently available in the transmission buffer. Similar to (1), we have

$$\hat{D}_G(s) = \sum_{l_{i,j} \in \mathcal{G}} d_{i,j}, - \sum_{l_{i,j} \in \mathcal{G}} \hat{a}_{i,j} d_{i,j}, \tag{2}$$

where $\hat{a}_{i,j}$ is the probability of whether the packet $l_{i,j}$ *can* be used for decoding,

$$\hat{a}_{i,j} = \prod_{l_{m,n} \in \mathcal{A}_{i,j}} (1 - p_{m,n}(s)). \tag{3}$$

Eq. (3) specifies that the probability, $\hat{a}_{i,j}$, is the product of the probabilities of the successful arrival of all its parent packets. $p_{m,n}(s)$ is the probability that packet $l_{m,n}$ (which may or may not belong to $\mathcal{G}$) is lost or delayed, for a given schedule s. Clearly $p_{m,n}$ depends on both the packet loss rate ϵ and the schedule s. For a given ϵ, if packet $l_{m,n}$ is scheduled to be transmitted earlier, it will have more chances to arrive on time, or to be retransmitted (within the delay constraint) if it is reported lost. We can formulate the scheduling problem as follows.

Given a set of packets $\mathcal{G}$, and given τ, ϵ, RTT, find the optimal schedule $s^* \in \mathcal{S}$, such that the expected playback distortion $\hat{D}_G$ is minimized, when all packets in $\mathcal{G}$ are transmitted at the time specified by s^*,

$$s^* = \mathbf{arg} \min_{s \in \mathcal{S}} \hat{D}_G(s), \tag{4}$$

where $\mathcal{S}$ is the set containing all possible schedules. Note that after the first packet in s^* is sent, the set $\mathcal{G}$ will change accordingly by removing this packet and possibly being filled with other new packets. Therefore s^* has to be re-computed after each packet is sent during the streaming session. In other words, the server is only interested in the *first* packet in s^* at any given time that a packet need to be selected for transmission. This property will be used in our proposed algorithm below.

IV. Proposed Scheduling Algorithm

The set of candidate packets $\mathcal{G}$ can contain packets of several frames, or the entire video sequence. For a streaming session where the source is captured in real-time, it is limited by the end-to-end delay τ. In general, we can use a "look-ahead" sliding window to specify which of the new packets should be considered as part of the set $\mathcal{G}$. Note that the candidate packet set $\mathcal{G}$ also includes the packets that are reported as lost and waiting for retransmission.

To find the optimal solution for our problem in (4), an exhaustive search over all candidate schedules is possible but may not be practically useful, since even a small candidate set $\mathcal{G}$ can leads to a large number of candidate schedules. The size of $\mathcal{S}$, and therefore the search complexity, increases exponentially with the size of $\mathcal{G}$. Exhaustive search is only applicable when the window size W is very small. Alternatives to an exhaustive search do exist as in the search algorithms proposed by Chou and Miao in [7], [8].

In this paper we propose a fast heuristic approach to solve this problem practically. The performance of the heuristic approach is very close to that of an exhaustive search (see experimental results below). At the end of this section we will discuss the difference between our approach and those in [7], [8].

A. Expected run-time packet distortion

Recall that only the first packet in s^* is actually used for transmission at a given time. Thus, instead of scheduling the order of the transmission of a packets in $\mathcal{G}$, if one can properly predict the importance of each packet in $\mathcal{G}$, then choosing the most "important" packet to be (re-)transmitted should provide a greedy solution to the scheduling problem.

In order to estimate the "importance" of a packet prior to the transmission, we propose the concept of *expected run-time distortion*, which take into account (i) the packet distortion $d_{i,j}$, (ii) the packet data dependencies, (iii) the channel conditions (e.g., loss rate, RTT), (iv) the packet playback time (deadline) and its delivery status (e.g., transmitted, received, lost, etc.).

We first introduce the concept of *run-time distortion*, $\hat{d}_{i,j}$, which enables us to capture the dependencies among packets (layers). We will show some examples before giving its definition. Consider transmitting or retransmitting a packet $l_{i,j}$. If it is *independent* from any other packets, its run-time distortion is equal to its original distortion, $\hat{d}_{i,j} = d_{i,j}$. If it has a child packet $l_{i,j+1}$, we will have following situations.
1. If $l_{i,j+1}$ has not been transmitted yet, transmitting $l_{i,j}$ only affects the layer itself.
2. If $l_{i,j+1}$ has been transmitted (but without any ACK or NAK), transmitting $l_{i,j}$ becomes more valuable since $l_{i,j+1}$ might be received and has to be decoded with $l_{i,j}$.
3. If $l_{i,j+1}$ has been transmitted and an ACK feedback is received, then $l_{i,j}$ becomes more important, because the received $l_{i,j+1}$ will be useless without this $l_{i,j}$.
4. If $l_{i,j+1}$ has been transmitted and a NAK was received we will have the same situation as in (1).
Similarly, the gain of transmitting (re-transmitting) $l_{i,j+1}$ also depends on the status of $l_{i,j}$. Extending the above to multiple layers, we define the run-time distortion of $l_{i,j}$ as follows:

$$\hat{d}_{i,j} = d_{i,j} \prod_{l_{m,n} \in \mathcal{A}_{i,j}} (1 - P^H(l_{m,n})) + \sum_{l_{m,n} \in \mathcal{B}_{i,j}} d_{m,n}(1 - P^H(l_{m,n})), \tag{5}$$

where $P^H(l_{m,n})$ is probability of loss/damage of that layer based on the transmission history of $l_{m,n}$, and is defined as

$$P^H(l_{m,n}) = \begin{cases} 1 & \text{if layer } l \text{ has not been sent,} \\ 1 & \text{if there is NAK on layer } l, \\ \epsilon^n & \text{if layer } l \text{ has been sent } n \text{ times,} \\ 0 & \text{if layer } l \text{ is ACKed.} \end{cases} \tag{6}$$

The term $d_{i,j} \prod_{l_{m,n} \in \mathcal{A}_{i,j}} (1 - P^H(l_{m,n}))$ in (5) shows that the original distortion of a packet is weighted by the probability of receiving all its parent packets. The second term, $\sum_{l_{m,n} \in \mathcal{B}_{i,j}} d_{m,n}(1-$

$P^H(l_{m,n})$), indicates that the importance of a packet increases if any of its children packets has been received. Before anything is transmitted, the run-time distortion of all layers is zero except for the lowest layer, for which the run-time distortion is equal to the original distortion $\hat{d}_{i,j} = d_{i,j}$. Eq. (5) implies that only after transmission (at least once) of all its parents, does a packet's run-time distortion become non-zero (except the base layer), and it increases if a child packet has arrived (or has been transmitted). Thus this definition reflects the "importance" of a layer with both data distortion and dependencies during the transmission.

Now we consider the lost probability of packet $l_{i,j}$ when we are able to send it at time $t^X_{i,j}$. We denote it as $P^X(l_{i,j})$ to distinguish it from P^H in (6). Since there could be possible retransmissions for a packet $l_{i,j}$ before its playback time t_i, we approximate $P^X(l_{i,j})$ as follows,

$$P^X(l_{i,j}) = \epsilon^{u_{i,j}}, \tag{7}$$

where $u_{i,j}$ is the number of possible retransmissions before packet $l_{i,j}$ passes its playback deadline, and can be obtained as

$$u_{i,j} = (t_i - t^X_{i,j} - t^D_{i,j})/RTT, \tag{8}$$

where $t^D_{i,j}$ is the transmission delay for $l_{i,j}$, i.e., $t^D_{i,j} = r_{i,j}/C(t)$, where $C(t)$ is channel bandwidth. Note that $C(t)$ can be varying over the time.

We here defined the *expected run-time distortion* $\widetilde{d_{i,j}}$ of a packet $l_{i,j}$ as follows,

$$\widetilde{d_{i,j}}(t^X_{i,j}) = P^X(l_{i,j}) \times \hat{d}_{i,j}, \tag{9}$$

where $\widetilde{d_{i,j}}$ is a function of $t^X_{i,j}$ since $P^X(l_{i,j})$ depends on it. We will use $\widetilde{d_{i,j}}$ as the importance metric of packet $l_{i,j}$.

B. Expected Run-time Distortion Based Scheduling – ERDBS

We now propose a fast scheduling algorithm to select packets for transmission at any given time. Denote t_{cur} as the current time at which the server wishes to select a packet to transmit. Note that t_{cur} corresponds to the time when the first packet in a schedule s is to be sent. We substitute $t^X_{i,j}$ in (9) by t_{cur}, for all the packets in $\mathcal{G}$, and select the packet with the largest value of $\widetilde{d_{i,j}}(t_{cur})$ to be transmitted at current time t_{cur}. The reason is that the expected run-time distortion evaluated at the current time t_{cur} reflects importance of the packet *if* it is to be transmitted at t_{cur}.

We refer to this approach as "Expected Run-time Distortion Based Scheduling" (ERDBS). The complexity is greatly reduced with this algorithm, since only one iteration for all the packets in transmission buffer is needed. The results in [2] shows that the packets selected using this greedy search, are almost identical to the ones selected by the optimal schedule in (4) using an exhaustive search.

C. Discussion

From (2) and (3), we can see that distortion decrease in $\hat{D}_G(s)$ by any single packet $l_{i,j}$ is heavily coupled with the transmission schedule of its parent packets. This introduces high complexity in the search for the optimal scheduling solution. As mentioned before, Chou and Miao [7], [8] already show that the dependencies of packets can be factored in a way such that the optimal schedule policy can be found with several iterations instead of a full search. The main difference between the approach in this paper and those in [7], [8] lies in the following two aspects. First, for any packet $l_{i,j}$, the status and distortion values of its child packets have been explicitly taken into account in our definition of the run-time distortion (see (5)). Second, the loss probabilities of all parent and child packets of $l_{i,j}$ are computed based *only* on the history of the transmission of those packets (see (6)), without any assumption of the future transmission schedules. In contrast, in [7], [8] these probabilities are calculated based on both the history and the transmission policies which are scheduled in the future.

Therefore, with (5) and (7) we are able to capture both the data dependencies and delay constraints into a single importance metric, i.e., *expected run-time distortion*, and the selection of packets can be done by choosing the packet with that maximum metric. Thus the full search in [6] or the iterative techniques in [7], [8] can be avoided.

V. Experimental Results

To evaluate our proposed schedule algorithm, in our simulation we also use a transmission scheme with no particular scheduling among different packets. We refer to this as the *Sequential Scheduling (SS)* scheme, in which the data packetization is the same as depicted in Section II-A. The packets within the same frame will be transmitted consecutively. The frames are delivered according to their original order in the encoded video stream. The SS scheme can discard a packet (either new or retransmitted) if it detects that this packet exceeds its playback time, in order to save bandwidth for other packets. The SS scheme selects proper layers of each frame to be sent to the client such that the average video data rate will not exceed the available channel bandwidth, and the remaining higher layers will be discarded.

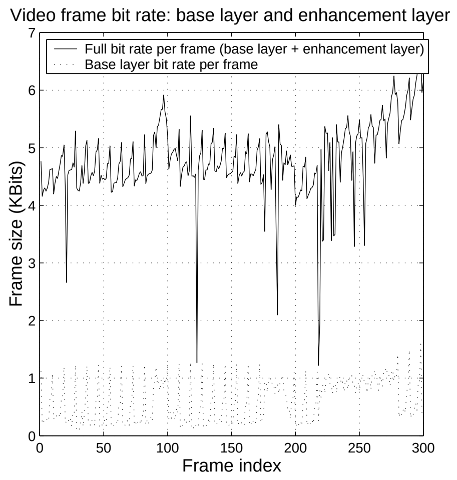
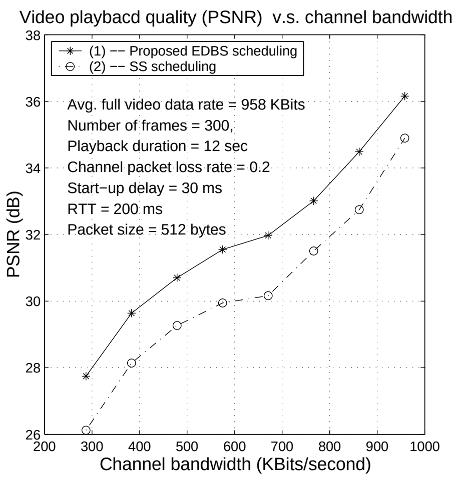


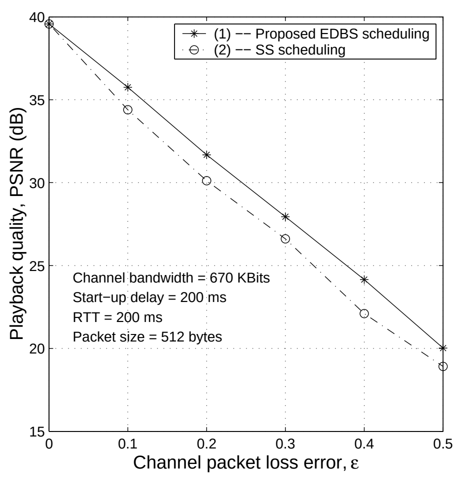
Fig. 3. MPEG 4 FGS frame data rate. The dotted line is the base layer size of each frame; and the solid line represents the full size of each frame (base layer and enhancement layer).

We use a MPEG 4 FGS video stream in our simulation. The video data is packetized with a fixed packet size of 512 bytes. The data used in the simulation is shown in Fig. V (we use the video sequence *Stefan* with 300 frames). The average video data rate after compression, with both base and enhancement layer, is 958 KBits/second. The channel packet loss rate is 0.2, i.e., 20% of the packets will be lost when transmitting over the server-client data channel, and packet loss is assumed to be i.i.d.. The round-trip-time is 200 ms and the streaming session has a start-up delay of 40 ms. The playback quality is measured by PSNR of the reconstructed video frames at the client end incorporating all the packets that were received on time. Over 100 realizations are averaged in the results we present.

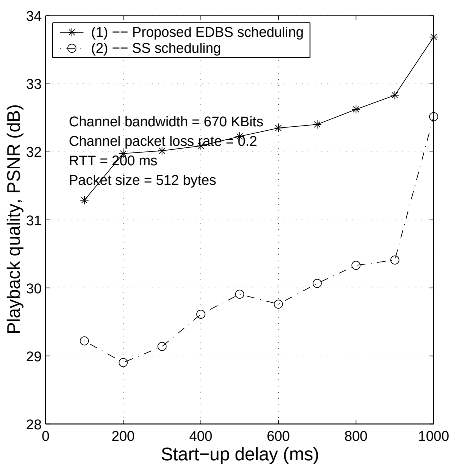
Fig. 4 shows the simulation results of the playback quality using the FGS video stream (in Fig. V) with various parameters (such as bandwidth, channel packet loss rate, start-up delay and RTT).



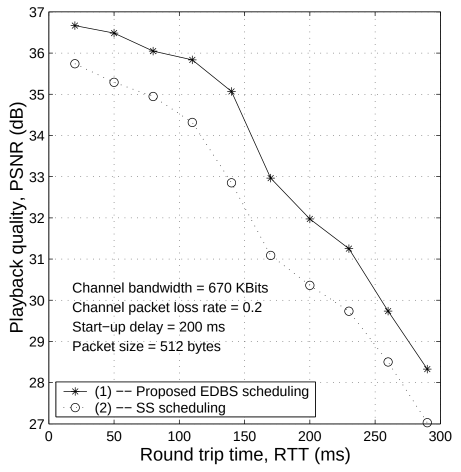
(a)



(b)



(c)



(d)

Fig. 4. The comparison of playback quality (PSNR) using proposed ERDBS and SS algorithm, in various conditions, such as bandwidth, channel packet loss rate, start-up delay and round trip time. In most cases the playback quality of ERDBS outperforms that of the regular SS delivery algorithm around 2 dB.

Our simulation shows that by using the proposed ERDBS algorithm to deliver the video data, the playback quality improves about 2 dB compared to that of using a SS delivery algorithm.

VI. Conclusions

In this paper, a new framework for delivery of scalable streaming media data over networks with packet loss is presented. We proposed a new delivery method, ERDBS, for this framework to solve the packet scheduling problem. The proposed algorithm, designed for a sender-driven transmission system, can increase the client playback quality by selecting proper packet(s) to be transmitted at any given time during the streaming session. The simulation results shows that ERDBS algorithms outperforms the other packet delivery schemes with a fixed scheduling, in the presence of channel packet loss. The low complexity of the search algorithm in ERDBS also enable it to be applied in real-time applications.

Acknowledgments

The authors would like to thank Philip A. Chou of Microsoft Research for very useful discussions and suggestions related to this work.

References

[1] "ISO/IEC JTC1/SC29/WG11, MPEG-4 version 2 visual working draft rev. 3.0, N2202," March 1998.

[2] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Proc. of 34th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, Novemeber 2001.

[3] C. Papadopoulos and G. Parulkar, "Retransmission-based error control for continuous media applications," in *Proc. NOSSDAV*, April 1996, pp. 5–12.

[4] M. Lucas, B. Dempsey, and A. Weaver, "MESH: distributed error recovery for multimedia streams in wide-area multicast networks," in *Proc. IEEE Int. Conf. on Commun.*, Montreal, Que., June 1997, vol. 2, pp. 1127–32.

[5] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable internet video using MPEG-4," *Signal Processing: Image Communication, 15 p.p. 95-126*, 1999.

[6] M. Podolsky, S. McCanne, and M. Vetterli, "Soft ARQ for layered streaming media," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, Special Issue on Multimedia Signal Processing, Kluwer Academic Publishers*, 2001, to appear.

[7] P. A. Chou and Z. Miao, "Rate-distortion optimized sender-driven streaming over best-effort networks," in *IEEE Workshop on Multimedia Signal Processing*, Cannes, France, October 2001, pp. 587–592.

[8] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," Tech. Rep. MSR-TR-2001-35, Microsoft Research Center, February 2001.

[9] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra, "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2000, IEEE Computer Society.

[10] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra, "Error control for receiver-driven layered multicast of audio and video," *IEEE Transactions on Multimedia*, 2001.

[11] C. Y. Hsu, A Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE JSAC, Special Issue On Multimedia Network Radios*, vol. 17, no. 5, pp. 756–773, May 1999.

[12] Z. Miao and A. Ortega, "Proxy caching for efficient video services over the internet," in *9th International Packet Video Workshop (PVW '99)*, New York, April 1999.

[13] Z. Miao and A. Ortega, "Rate control algorithms for video storage on disk based video servers," in *Proc. of 32nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 1998.