Optimal Scheduling for Streaming 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 appears to be more suitable for transmission over the current heterogeneous networks. In this paper we study the problem of scalable layered streaming media delivery over a lossy channel. The goal is to find an optimal transmission policy to achieve the best playback quality at the client end. The problem involves some trade-offs such as time-constrained delivery and data dependencies. For example, a layer should be dropped before transmission if it already has a delay such that it cannot be played before its scheduled time. Moreover, less important layers with near-playback-time may also be dropped or delayed for delivery in order to save bandwidth for other layers with a high priority. We propose a framework for scalable streaming media delivery, that involves a novel scheduling algorithm called *Expected runtime Distortion Based Scheduling*, EDBS, which decides the order in which packets should be transmitted in order to improve client playback quality in the presence of channel losses. A fast greedy search algorithm is presented that achieves almost the same performance as an exhaustive search technique (98% of the time it results in the same schedule) with very low complexity and is applicable for real-time application.

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.

I. Introduction

Streaming media applications are becoming increasingly popular on the Internet. Usually the term "streaming" means that the real-time data, e.g., audio or video, is delivered from server to client continuously, i.e., the client starts to playback a video and/or audio sequence after a certain initial delay before the entire sequence has been fully downloaded. Streaming media data is constrained by strict delay bounds. In the presence of network congestion or channel error, the successful on-time delivery cannot be guaranteed since the retransmission of lost packets is limited due to delay constraint. However, unlike other data, it can still be useful even with some loss. Therefore, the delivery of media data should be treated differently from other traditional data. The heterogeneity of the Internet results in bandwidths that are different in different parts of the network and can also vary over time. For this reason scalable media representations are very useful. A stream in scalable representation has several layers, from low (or base) layers to high (or enhancement) layers. Decoding with only lower layers results in a coarse version of the reconstructed signal, while adding more layers can enhance the reconstructed quality. However, higher layers are not decodable unless all the corresponding lower layers are available to the decoder. Thus, scalable media has the advantage of enabling data transmission over channels with different bandwidth, and being robust under bad network conditions, e.g., congestion, damage, delay or loss, since in these network conditions, higher (less important) layers can be dropped to provide more chance for lower (more important) layers to arrive on time for decoding.

In this paper, we focus on the problem of scalable streaming media end to end delivery over a lossy channel. In particular, we consider mechanisms to schedule the packets to be transmitted. Consider a simple example. Suppose there are 5 packets in the sender's buffer, each representing one layer of media data. These packets contain layers 1 and 2, $L_{i,1}, L_{i,2}; L_{i+1,1}, L_{i+1,2}$, of frames i and $i+1$, respectively; and layer 1 of frame $i-1$, $L_{i-1,1}$ which is indicated to be lost and has to be retransmitted. There can be many transmission schedules available for those packets, some possible examples are listed in Table.-I. Schedule (1) is a straightforward approach, it retransmits $L_{i-1,1}$ first, then sends all layers of frame i followed by all layers of frame $i+1$. Schedule (2)

Schedule 1	Schedule 2	Schedule 3	...
(Re-tx) $L_{i-1,1}$	$L_{i,1}$	$L_{i,1}$	...
$L_{i,1}$	$L_{i,2}$	$L_{i+1,1}$	...
$L_{i,2}$	$L_{i+1,1}$	$L_{i,2}$	...
$L_{i+1,1}$	$L_{i+1,2}$	$L_{i+1,2}$	...
$L_{i+1,2}$	(Drop) $L_{i-1,1}$	(Drop) $L_{i-1,1}$	...

TABLE I

TRANSMISSION SCHEDULES. EACH COLUMN REPRESENTS A POSSIBLE SCHEDULE OF DELIVERING PACKETS BY THE SEVER. THE ORDER OF DELIVERY IS FROM THE TOP TO BOTTOM ON EACH COLUMN

decides to cancel retransmission, since $L_{i-1,1}$ may not arrive on time for the playback of frame $i-1$, and because canceling this retransmission may help other layers to arrive on time. Schedule (3) switches the delivery order of layers in frame i and $i+1$, sending all lower layers first followed by all higher layers. In case of bad network conditions, the two lower layers will have a good chance to arrive on time. Since higher layers are useless without the presence of *all* corresponding lower layers, Schedule (3) can help to improve the playback quality since it is more likely that lower layers arrive at the decoder on time. Obviously there are other possible choices of schedules for this example. Our goal in this paper is to determine what scheduling mechanisms provide the best playback quality. The schedule, i.e., the order for packet delivery that is best for a given situation, highly depends on many factors, such as the media content, the data dependency among the scalable formats, the playback delay constraints, and channel conditions. In this paper we will present a framework to solve this problem.

In our scheme, we estimate the "importance" of all packets in the transmission buffer, including both the new packets and those requesting to be retransmitted. We introduce the concept of "expected runtime distortion" in this model to predict how each packet affects the receiver's playback quality in a scalable streaming media with complex data dependencies and channel

conditions. We provide a fast greedy search algorithm as well as an exhaustive search method. Both of them achieve almost the same performance, but the greedy search algorithm has very low complexity and can be implemented in real-time easily. Our simulation shows the improvement of playback quality with both the greedy and exhaustive methods. We test our scheme in various conditions, such as round trip time, initial playback delay, channel loss rate, etc..

A. Background and related work

The delay constraint of real time streaming introduces limitations on the server retransmission capability. Assuming the server gets feedback from the client, i.e., acknowledgments (ACKs) or negative acknowledgments (NAKs). The lost packets (indicated by a NAK), might not arrive to the receiver on time if retransmission is needed, especially if the round-trip-time (RTT) is high and channel error is large. The initial play delay of streaming media delivery *does* allow limited retransmission. For example, in [1], [2], 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.

Other research has addressed the delivery scalable streaming media (e.g., MPEG-4 based scalable video coding). In [3] 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, which are necessary for decoding the corresponding enhancement 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 complement work for other rate-controlled based scheme which decides the rate of the video bit-stream to be transmitted, e.g., [3].

Podolsky *et al* [4] have studied policies for scheduling of scalable media data. In their work, a

Markov chain analysis is used to estimate the average distortion of each candidate policy. For each set of system parameters an optimal steady state policy is found. This design is performed off-line. On-line adaptation between different policies is achieved by estimating the system parameters and switching to the policy that is most suitable. The main difference in our work is that no attempt is made to define deterministic policies and rather a simple algorithm is used to determine on-line which packet should be sent next. The fact that policies do not have to be defined in advance allows more flexibility, e.g., in our approach we can use the exact distortion values for each frame, instead of making the assumption these are always the same as in [4].

Chou *et al* [5], [6] have also addressed similar issues. In [6] the data dependencies' effect on distortion is taken into account in a manner similar to the one proposed here, although in a multicast scenario. Also the problem we address here is studied in [5] where a tree-structure recursive algorithm for searching the optimal schedule is proposed. The main differences with respect to this work is that we use a simpler algorithm to find the schedule (we use a greedy technique) as well as a simplified distortion metric.

The paper is organized as follows. Section II provides an overview of a scalable streaming media delivery system. Section III formulates our scheduling problem and presents solutions based on both exhaustive and greedy search methods. Section IV shows the simulation results and Section V concludes the paper.

II. Problem Setting

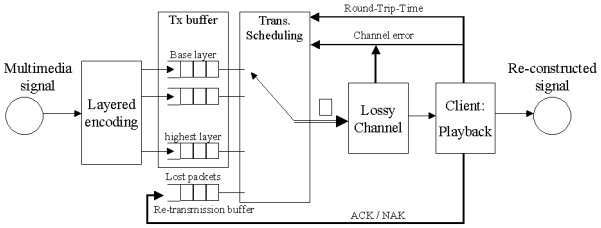

Fig. 1. System architecture

A. System Architecture

Fig.-1 shows the architecture of our proposed system. The media sequence is segmented into frames, which are compressed into several layers. The layers are packetized and fed into the server's transmission buffer, these are the "new" packets waiting to be scheduled for transmission. There are also packets in the server's buffer that are waiting for retransmission because the client reported them lost. The server's scheduling module selects one packet at a time from those buffers and sends it to the lossy channel. Some packets can be lost, damaged or delayed (delayed packets are also considered lost if they exceed their playback delay). At the client end, the lost or damaged packets are reported to the server via a feedback channel. Information related to Round-Trip-Time (RTT) and channel error can be retrieved by server via client's feedback.

The scheduling module is the crucial part of the whole system, its main tasks are: (1) Determine how many layers of each frame have to be delivered. This can be achieved by rate control methods presented in [7], [8] and is beyond the scope of this paper. We assume the number of layers to be used for each frame (which may be different for each frame) is pre-determined before transmission.

(2) Estimate the "importance" of packets in the transmission buffer, which is measured by our proposed *expected run-time distortion* (introduced in Sec-III-C.1). The procedure predicts each packet's value to the playback quality before they are transmitted and takes into account data distortion, data dependency, delay constraint, channel error, round-trip time, rate budget, receiver feedback, etc.

(3) Choose the delivery order. The scheduler runs the algorithms proposed in this paper to select the order in which packets should be transmitted, choosing between packets containing new layers data and packets to be retransmitted based on their expected run-time distortion.

B. Parameters and definitions

We assume a simplified binary erasure channel (BEC) in our paper. Let ϵ be the probability a packet is lost. We assume that ϵ is estimated from information provided by the client on observed packet losses, and could be time varying.

The 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. Therefore there can be more time for the server to re-send a packet if necessary before its time-out. RTT can be also be used by the sender as the "time-out" threshold in the case of lost of feedback.

The streaming media playback encounters an initial delay or latency before playback. Usually a larger initial delay can smooth out more variations in channel capabilities, and enable more time for retransmissions, resulting in a better playback quality. However, the receiver might prefer a smaller latency before playback and therefore there is trade-off between the initial delay and playback quality. The readers are referred to [7], [9], [8] for more details.

C. Scalable media sequence

In this paper, we assume a scalable (layered) encoder is available at server end, or the streaming media data is pre-compressed in a scalable format, for example a scalable MPEG-4 format [3]. Besides the dependencies among layers within each frame, there are also dependencies between consecutive frames. Many current video compression standards use motion estimations among consecutive frames, e.g, the I, P, B frames in the MPEG-2 standards P frames can not be decoded unless the corresponding I frames are received; while B frames can not be decoded until all corresponding P (or I) frames are decoded. We refer to all those dependencies across frames and layers as *data dependencies.* While our experiments are based on audio data that has simpler dependencies (i.e., there are no dependencies across frames), the techniques we used can be easily extended to more complex dependencies such as those found in MPEG video.

A traditional way of measuring distortion in multimedia signal (i.e., image video) is using PSNR (Peak Signal to Noise Ratio). Reconstruction with only base layer results in a higher distortion, i.e., a smaller PSNR; while with as more layers are used the distortion is reduced. In our framework, we use a "relative" distortion measurement of media quality. We define the *total distortion* of a frame, D_f, as the distortion when a frame is completely lost; a frame reconstructed with all its

layers has a minimum distortion, i.e., zero. Define the total distortion of the media stream as the sum of D_f 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 decoded at playback. For example, in a media sequence with N frames, each frame has L layers, where $L_{i,j}$ (the j^{th} layer of the i^{th} frame) has a distortion $d_{i,j}$, and $a_{i,j}$ is an indicator defined as $a_{i,j} = 1$ if $L_{i,j}$ is *used* for playback, otherwise $a_{i,j} = 0$. The playback distortion is defined as

$$D_a = \sum_{i=1}^{N}\sum_{j=1}^{L} d_{i,j} - \sum_{i=1}^{N}\sum_{j=1}^{L} a_{i,j}d_{i,j} = \sum_{i=1}^{N}\sum_{j=1}^{L}(1 - a_{i,j})d_{i,j} \tag{1}$$

A practical example of this definition will be presented in detail in Section-IV.

III. Problem formulation and proposed algorithms

A. Problem formulation

Since the schedule should be made before the transmission, the sender has to predict the playback distortion when making a schedule to reduce it. We call the distortion predicted by the server before transmission the "*estimated distortion*", $\widetilde{D_a}$.

The scheduling problem can be summarized as follows. For a group of packets in a sender's buffer, $\mathbf{G}$, (for simplicity, assume each packet contains one layer, $L_{i,j}$), given the initial delay T_I, channel loss rate ϵ, round-trip-time RTT, feedback information (such as ACKs, NAKs), there is a set of possible orders, $\mathbf{S}$, for sending packets in $\mathbf{G}$ (e.g., the schedules in Table-I). The goal is to find the optimal order, denoted as a schedule $s^* \in \mathbf{S}$, to send packets $L_{i,j} \in \mathbf{G}$ which minimizes the estimated playback distortion $\widetilde{D_a}$,

$$s^* = \mathbf{arg}\min_{s\in\mathbf{S}} \widetilde{D_a} \tag{2}$$

Since the parameters of the system are time-varying in this paper we focus on finding "local" optimal schedules, i.e., $s^*(t_i)$ for each time t_i, which operate based on current observed conditions. $s^*(t_i)$ is only valid for the first packet scheduled to be sent and is re-calculated after that. This can also be regarded as a *transmission policy* π, selecting a *proper* packet to send from buffer at each transmission time, to minimize the playback distortion.

Obviously a good estimated distortion, $\widetilde{D_a}$, should be obtained since we base our scheduling on minimizing this distortion. Thus providing a good estimate will be a focus of our work.

B. Sequential scheduling – SS

One easy way for selecting packets to send, denoted here as *Sequential Scheduling (SS)*, consists of sending frames according to the original frame sequence, with layers sent in order of importance within each frame. Thus, we send all layers of frame i, followed by all layers of next frame $i+1$, etc. We assume SS will discard a packet (either new or retransmitted) before transmission if it detects that this packet exceeds its playback time.

C. Expected Distortion Based Scheduling – EDBS

C.1 Expected run-time distortion

The run-time expected distortion $\widetilde{d_{i,j}}$ of layer $L_{i,j}$ is defined as

$$\widetilde{d_{i,j}} = \widetilde{p_{i,j}} \times \hat{d}_{i,j}, \tag{3}$$

where $\widetilde{p_{i,j}}$ is the *expected loss probability* of layer $L_{i,j}$ and $\hat{d}_{i,j}$ is the *run time distortion* of that layer. The sender estimates $\widetilde{p_{i,j}}$ before transmission given by the channel loss rate ϵ and its possible number of retransmission attempts $A_{i,j}$.

$$\widetilde{p_{i,j}} = \epsilon^{A_{i,j}}. \tag{4}$$

$A_{i,j}$ depends on the channel round-trip-time RTT, and the packet life-time T_L,

$$A_{i,j} = T_L/RTT. \tag{5}$$

$$\text{where} \qquad T_L = T_P - T_X - T_D. \tag{6}$$

T_X is the starting transmission time of $L_{i,j}$, T_D is the transmission delay for $L_{i,j}$, i.e., $T_D = r_{i,j}/C$, where $r_{i,j}$ is the size of the layer and C is channel transmission rate.

The *run-time distortion*, $\hat{d}_{i,j}$ in (3), is used to resolve the dependencies between layers. If a packet is *independent* from any other packets, its run-time distortion is equal to its original distortion, $\hat{d}_{i,j} = d_{i,j}$. Given the dependencies among frames and layers, since higher layers are not decodable without the presence of the necessary lower layers, the lower layers should incorporate this factor into their importance, besides their original signal distortion. For example, consider the retransmission request of a layer $L_{i,j}$ depending on the status of its corresponding higher layer $L_{i,j+1}$:

1. If $L_{i,j+1}$ has not been transmitted yet, retransmitting $L_{i,j}$ only affects the layer itself;

2. If $L_{i,j+1}$ has been transmitted but without any ACK/NAK, retransmitting $L_{i,j}$ becomes more valuable since $L_{i,j+1}$ might be received cannot be decoded without $L_{i,j}$;

3. If $L_{i,j+1}$ has been transmitted and an ACK received, then even more value is attached to $L_{i,j}$, since without it, the already received $L_{i,j+1}$ would be useless.

4. If $L_{i,j+1}$ has been transmitted and a NAK was received we are in the same situation as in (1).

Similarly transmitting (re-transmitting) $L_{i,j+1}$ also results in different gain depending on the status of $L_{i,j}$. These dependencies can be extended to multiple layers. We define all the layers that $L_{i,j}$ depends on as the *parents* of layer $L_{i,j}$, and group them in a set $\mathcal{A}_{i,j}$. The layers that depend on $L_{i,j}$ are its children, and are grouped in a set $\mathcal{B}_{i,j}$. The run-time distortion of $L_{i,j}$ (denoted as $\hat{d}_{i,j}$) is defined as follows:

$$\hat{d}_{i,j} = d_{i,j} \prod_{l \in \mathcal{A}_{i,j}} (1 - P_{loss}(l)) + \sum_{l \in \mathcal{B}_{i,j}} d(l)(1 - P_{loss}(l)), \tag{7}$$

where l is an element (a layer) in set $\mathcal{A}_{i,j}$ or $\mathcal{B}_{i,j}$, $d(l)$ is the original distortion of layer l, and $P_{loss}(l)$ is probability of loss/damage of that layer, and is defined as

$$P_{loss}(l) = \begin{cases} 1 & \text{if layer } l \text{ has not been sent} \\ 1 & \text{if there is NAK on layer } l \\ \epsilon^n & \text{if layer } l \text{ has been sent } n \text{ times} \\ 0 & \text{if layer } l \text{ is ACKed} \end{cases} \tag{8}$$

The term $d_{i,j} \prod_{l \in \mathcal{A}_{i,j}} (1 - P_{loss}(l))$ in (7) shows that the original distortion of a layer is weighted

by the probability of receiving all its parent layers. The second term $\sum_{l \in \mathcal{B}_{i,j}} d(l)(1 - P_{loss}(l))$ indicates that the importance of a layer increases if any of its children layers 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 $\hat{d}_{i,j} = d_{i,j}$. (7) implies that only after transmission (at least once) of all its parents, does a layer's run-time distortion become non-zero (except the lowest layer), and it increases if a child layer has arrived (or is being transmitted). Thus this definition reflects the "importance" of a layer with both data distortion and dependencies during the transmission. A similar approach to defining the distortion of a layer/packet and resolving the data dependencies is also being developed in [5]

C.2 Expected distortion for packets in the transmission buffer

In a real-time streaming application, it is not possible for a sender to examine the entire media sequence if the sender only has limited buffer. Thus the sender can only make the policy based on the packets currently in the buffer. Accordingly, we modify (2) as follows,

$$s^* = \mathbf{arg} \min_{s \in \mathbf{S}} \widetilde{D_b}(s) \tag{9}$$

Where $\widetilde{D_b}$ indicates the estimated distortion of all packets in the sender's buffer, which is the sum of run time expected distortion of those packets.

$$\widetilde{D_b} = \sum \widetilde{d_{i,j}}, \quad \text{for all packets in the buffer} \tag{10}$$

As described in Sec.-III, $\widetilde{D_b}$ is only the estimation of real playback distortion. Minimizing $\widetilde{D_b}$ does not necessary imply the best playback quality is achieved, though we already consider all the possible parameters and trade-offs in the estimation. To verify the effectiveness the estimation method, the corresponding simulation (Section-IV) is based on the actual receiver's playback distortion (the packets actually received) for our proposed scheduling algorithm of transmission policy.

A few variables need to be updated to calculate the total estimated distortion $\widetilde{D_b}$. Combining

(3), (4), (5) and (6), we re-write the run time expected distortion $\widetilde{d_{i,j}}$ as,

$$\widetilde{d}_{i,j} = \tilde{p_{i,j}} \times \hat{d}_{i,j} = [\epsilon^{\frac{(T_{P_{i,j}} - T_{X_{i,j}} - T_{D_{i,j}})}{RTT}}] d_{i,j} \tag{11}$$

For a possible schedule s, the starting transmission time of each layer $L_{i,j}$ is assigned by this schedule. However only the first scheduled packet will be transmitted at the time as it is scheduled, the following packets might be delayed by any possible retransmissions of the previous lost packets. To approximate this possible delay caused by retransmissions, we define the expected transmission delay $\widetilde{T_{D_{i,j}}}$ as,

$$\widetilde{T_{D}}_{i,j} = \frac{r_{i,j}}{C/(1-\epsilon)} = \frac{T_{D_{i,j}}}{1-\epsilon}. \tag{12}$$

The starting transmission time of a packet in a schedule s is set to be the current time t_c plus all the expected transmission delay of packets before it,

$$\widetilde{T_{X}}_{i,j} = t_c + \sum_{(k,m)<(i,j)} \widetilde{T_{D}}_{k,m}. \tag{13}$$

The notation "$(k,m) < (i,j)$" means that layer $L_{k,m}$ is scheduled before layer $L_{i,j}$ to be transmitted. Updating (11) with $\widetilde{T_{D_{i,j}}}$, $\widetilde{T_{X_{i,j}}}$, we can calculate the total estimated distortion D_b of all packets in the buffer with respect to a specific schedule s. The optimal schedule s^* is chosen with the minimum D_b.

C.3 Exhaustive and greedy search

The set of all possible schedules $\mathbf{S}$, includes all the combination of the layers in the transmission buffer. When the buffer size is large, the large number of combinations makes finding the optimal solution via exhaustive search infeasible in practice. However, we use exhaustive search as a benchmark for the simulation. We also provide an heuristic algorithm called *greedy search* that can be used in real-time with low complexity. Instead of using the cost function (2), which implies a large set of possible schedules, the greedy search is based on a cost function which considers the run time expected distortion of *individual* packets. The basic idea here is try to identify the *most important* packet at current time according to all current parameters and transmission/playback

status without going through all the possible schedules. To "quantify" the importance of each packet in the transmission buffer at current time t_c, we calculate the run time expected distortion of that packet as if it is scheduled to be sent at time t_c. Then we choose the most important one, which is the one with the *maximum distortion* to send for time t_c. Since only one iteration for all the packets in transmission buffer is performed, the complexity is greatly reduced. The simulation results show that the packets selected using this greedy search, is almost identical to the one selected by the optimal schedule in (2) found by exhaustive search.

IV. Experimental results

We use a layered encoded audio streaming[1] for our simulation. The audio stream is segmented into "frames". Each frame lasts 750 ms and is encoded into 12 layers, denoted as layer 1 (lowest layer) to layer 12 (highest layer), each layer contains 512 bytes. The decoding of a layer (from layer 2 to layer 12) depends on all the layers lower than it, while layer 1 can be decoded by itself. There are no dependencies across the frames. The distortion is measured in Mean Square Error (MSE). A typical frame distortion is shown in Table-II. Upon receiving a layer, the MSE is reduced for the reconstructed frame (as in column 2). This represents the quality "improvement" value of this layer. We assume there is no error concealment used in simulation of all delivery methods. A lost frame (missing the lowest layer) is replaced by all zero values. The distortion is reduced to zero after receiving all layers. In the simulation, we compare the playback distortion between our proposed delivery algorithm (EDBS) and SS delivery. The playback distortion is obtained at receiver end by calculating the distortion of each reconstructed frame with the layers available. The distortion of the entire stream is the sum of the distortion of individual frames and is normalized. Thus a zero playback distortion means all packets are received on time, and distortion equal to 1 means all frames are not available for playback. The 12 layers produce 6K bytes per 0.75 second, or 64 Kbits/sec data rate. The average channel rate in the simulations is set to be 64 KB/sec to accommodate all the layers. with the loss rate ϵ varies from 0.05 to 0.6 in our simulations.

[1]Developed by Microsoft Research.

Layer No.	Reduced Dist.	Remaining Dist.
No layers	0.0	52304.3
1	27201.0	25103.3
2	9963.2	15140.0
3	4888.8	10251.2
4	3767.2	6484.0
5	1399.8	5084.2
6	1501.4	3582.8
7	1501.4	2081.3
8	683.9	1397.5
9	372.8	1024.7
10	469.3	555.3
11	469.3	86.0
12	86.0	0.0

TABLE II

THE AUDIO DATA DISTORTION IN ONE FRAME

Fig.-2 shows the comparison between SS and EDBS. With the SS scheme, the playback distortion (normalized) remains fairly constant independently of the buffer size. While the performance of EDBS improves a lot when the buffer size increases. The performance improvement for EDBS is limited after the transmission buffer exceeds a certain size, e.g., 6 KBytes in this case. Thus only limited packets in the buffer needs to be examined in the scheduling procedure, even with a presence of a larger physical buffer size, to achieve its best performance, so that the computation complexity can can be reduced. Figs.-3, 4 and 5 show the comparison between the two algorithms with different parameters, RTT, playback delay and channel loss rate, respectively. EDBS outperforms SS in all the situations.

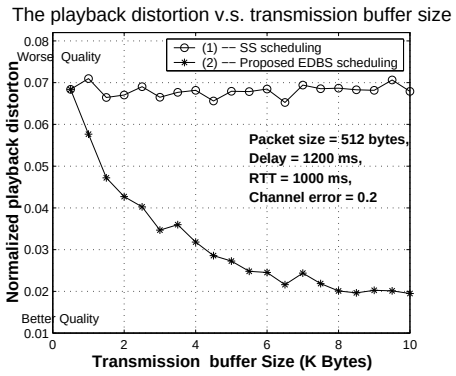

Fig. 2. Playback distortion with respect to different sender's buffer size. The channel error is 20% loss rate. With EDBS scheduling algorithm, the playback distortion decrease very fast when the buffer size increases. EDBS performance greedy search algorithm over all the packets in the sender's buffer.

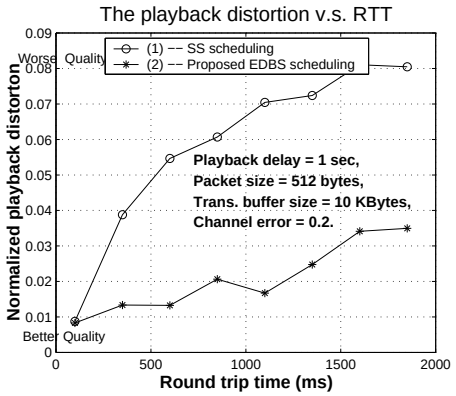

Fig. 3. Playback distortion with respect to various RTT. With different round-trip-time, from 500 ms second to 4 second.

For all the experiments on EDBS algorithm, we use both the exhaustive search and the greedy search methods. Only the curves for greedy search are plotted, since both of methods produces almost identical playback distortion and curves are overlapped if plotted together. Our experiments shows that up to 98% of the policy decision made (the packet selected) during the whole sequence transmission are same from both method. The playback distortion for all the results is obtained after averaging more than 500 realizations.

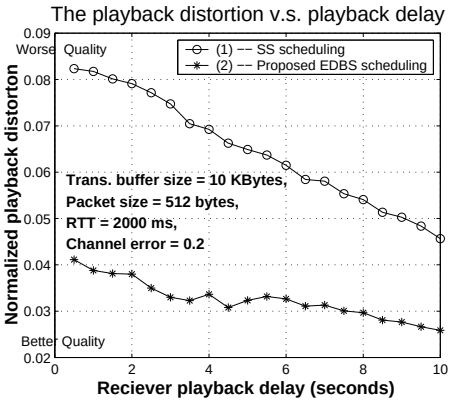

Fig. 4. Playback distortion with respect to various initial playback delay

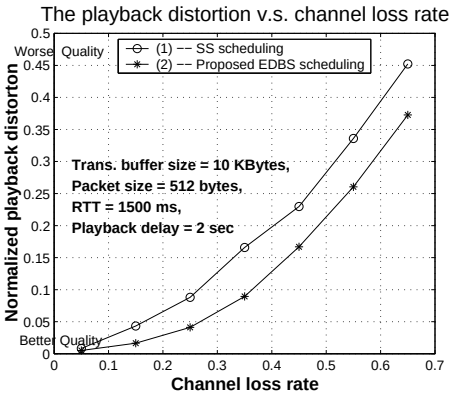

Fig. 5. Playback distortion with respect to various channel error

V. Conclusions

In this paper, a new framework for delivery scalable streaming media data over networks is presented. We proposed a new delivery method, EDBS, in this framework to solve the scheduling problem of the packets of scalable streaming media in the server buffer before transmission. We provide a fast algorithm, greedy search, which has almost the same performance (up to 98%) as an exhaustive search. The simulation results shows that EDBS algorithms outperforms the traditional transmission methods in various situations, with different RTT, playback delay, channel error, etc.. The low complexity of the greedy search algorithm for EDBS also enable this framework running in real-time applications with losing only 1% of the performance compared to using an exhaustive search.

Acknowledgments

The authors would like to thank Phil Chou of Microsoft Research for his generosity on providing the scalable layered compressed audio streaming data for the simulation in this paper and very useful discussions.

References

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

[2] 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.

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

[4] 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*, April 2000.

[5] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," 2000, In preparation.

[6] 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.

[7] 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*, 1999.

[8] 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.

[9] 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.