Rate-Distortion Optimized Scheduling for Redundant Video Representations

Huisheng Wang and Antonio Ortega

Abstract

This paper extends rate-distortion optimized streaming techniques to operate on a general class of coding formats that explicitly support redundancy in their coding structure. Examples include multiple description layered coding (MDLC) and multiple independently encoded versions of a video source. Such source codecs usually produce multiple decoding paths, while previous work on video streaming has mostly focused on those encoding techniques that only generate a single decoding path. A new source model called Directed Acyclic HyperGraph is introduced to describe the dependency and redundancy relationship between different video data units with multiple decoding paths. Based on this model, we then propose two rate-distortion based packet scheduling algorithms, i.e., Lagrangian optimization and a greedy algorithm, to dynamically adjust the system's real-time redundancy to match the channel behavior. The proposed streaming system introduces two types of redundancies, namely, source redundancy and transport redundancy. This paper presents a detailed performance analysis of the individual benefits for error robustness provided by these redundancies and their interplay. Experimental results show that our proposed system with both redundancies achieves the best end-to-end performance on real-time video communication over a wide range of network scenarios.

I. Introduction

Internet multimedia applications, such as live video streaming, distance learning and video on demand services, are becoming increasingly popular. Given the best-effort service offered by the current Internet, video transmission is inevitably affected by the network variations in bandwidth, delay and packet loss rate, and thus it is imperative to provide some means to deal with the transmission impairments.

A variety of techniques have been proposed in the literature to address error control, including forward error correction, delay-constrained retransmission [1], intra/inter mode switching [2], reference picture

H. Wang and A. Ortega are with the Signal and Image Processing Institute and Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564. Email: {huisheng.wang@gmail.com, ortega@sipi.usc.edu}.

selection [3,4], dynamic packet dependency control [5], layered coding with unequal error protection [6], soft ARQ for layered streaming media [7], and multiple description coding [8,9]. An important recent advance to video streaming is the rate-distortion optimized packet scheduling (RaDiO) framework initially proposed by Chou and Miao [10,11]. This work formalized packet dependencies as a directed acyclic graph (DAG), prioritized packets based on their importance, and scheduled them so as to minimize a Lagrangian cost function combining expected distortion and rate. Some techniques have been proposed to reduce the complexity of the original algorithm. Miao and Ortega [12,13] simplified the approach by running a greedy algorithm that explicitly combines the effects of data dependencies and delay constraints into a single importance metric. Chou and Sehgal [14] presented simplified methods to approximate the optimized policies. Chakareski *et al*. [15] proposed a family of simplified distortion models to approximate the end-to-end distortion produced by arbitrary packet loss patterns. Recent work by De Vleeschouwer *et al*. [16] improved the performance of greedy scheduling algorithm by delaying packet scheduling decisions to avoid premature retransmissions. The sender-driven rate-distortion framework in [10] has also been extended into other transmission scenarios, including packet scheduling at the receiver [14], at an intermediate proxy [17], or taking into consideration path diversity [18].

As an alternative to traditional error-resilient encoding techniques that introduce redundancy at the bit stream level, this paper extends the RaDiO framework proposed in [10,11] to operate on a general class of coding formats that explicitly support redundancy in their coding structure by, for example, producing multiple redundant representations of the video content. Previous work on RaDiO is mainly focused on encoding techniques, such as layered coding [19], which generate packets that can only be decoded following a *single decoding path*: a packet can be decoded *only* when all the packets it depends on are received and decodable. However, source codecs that explicitly support redundancy to combat transmission errors usually produce *multiple decoding paths*: there are multiple ways to decode a packet, each with a different distortion reduction depending on which packets, among those it depends on, are received. One example of these codecs is multiple description layered coding (MDLC) proposed in [20], which combines the hierarchical scalability of layered coding (LC) with the reliability of multiple description coding (MDC). Other coding examples with multiple decoding paths include multiple independent encodings or decoding with error concealment. Note that in the absence of adaptation the redundancy levels may not match those required by the actual network conditions. Here we propose an on-line rate-distortion based scheduling algorithm that can dynamically adjust the system's real-time redundancy to match the channel behavior so as to achieve better overall quality.

The scheduling problem becomes more challenging when considering multiple decoding paths. In

addition to challenges arising in [10, 11], such as delay constrained delivery, channel conditions and data dependency, the scheduling algorithm has to take into account the correlation or redundancy between data units for end-to-end distortion estimation. The basic RaDiO framework [10, 11] used a simple DAG model to represent data dependency only, and is thus limited to coding scenarios that have a single decoding path. Cheung and Tan [21] introduced a more general formulation based on the DAG model to include the case where a packet can be decoded in different ways. They considered all possibilities of decoding and delivery scenarios, which leads to significant increases in complexity. In our approach, we propose a new source model that introduces additional components on top of a DAG, in order to explicitly represent source redundancy among packets. Thus, compared to [21], our approach provides a more systematic way to represent source codecs that support multiple decoding paths with reasonable complexity. Part of the concepts of this source model and a heuristic scheduling algorithm tailored to a simplified MDLC codec with only I-frames have been proposed in our previous work [20]. A preliminary version of our general streaming framework was presented in [22].

There are two main contributions in our research. First, we propose to optimize the level of redundancy added to a stream by *jointly* designing both transport and source coding redundancy mechanisms, further investigating the interplay between these two different types of redundancies. Thus, in our work, enhanced adaptation flexibility is achieved by combining i) an encoding algorithm that supports various levels of source redundancy in it coding structure and ii) a transport mechanism that dynamically adjusts the run-time redundancy of the compressed bit streams during transmission by applying rate-distortion optimized packet scheduling. Previously proposed techniques [23, 24] optimized source redundancy only at the encoding stage and had limited flexibility at runtime. Thus a fixed combination of LC and MDC was chosen and received video quality could be significantly degraded if the assumptions made at encode time about the network state proved to be incorrect. In contrast, our work provides more flexibility by allowing run-time redundancy control to deal with a larger range of network conditions.

Second, unlike techniques that address a particular source coding approach [18, 25], we propose a general framework by formalizing a more structured source model that can represent various source coding approaches. Given appropriate source model parameters defined in Section III-C, our algorithm can be directly applied to various coding scenarios, including multiple independent encodings [25] and decoding with error concealment [18]. Compared to stream switching often used in commercial streaming systems [26, 27], our approach enables finer switching, i.e., at the packet level rather than the stream level, and further allows more flexible adaptation options than simple switching. In addition, we also introduce an improved MDLC predictive coder based on previously proposed MDC codecs [28–30] and

use it to evaluate the performance of our proposed scheduling algorithms under various redundancies for a number of video sequences.

Specifically, we first propose a new Directed Acyclic HyperGraph (DAHG) source model to represent both data dependency and correlation between different video data units. The DAHG model introduces the concepts of multiple decodable states and multiple decoding paths, from which the expected end-to-end distortion D for a group of packets can be estimated accurately under a vector of packet loss probabilities, $\boldsymbol{\epsilon}$, for each packet in the group. In addition, a Taylor series expansion of D in terms of $\boldsymbol{\epsilon}$ reveals important properties for different coding scenarios, depending on whether source redundancy exists or not. We then propose two rate-distortion adaptive packet scheduling algorithms, one based on Lagrangian optimization with the iterative descent approach proposed in [10] and another one based on a greedy solution derived from the Taylor expansion.

It is noted that, in addition to source redundancy explicitly produced at the encoding stage, the proposed streaming framework implicitly introduces a transport redundancy by allowing retransmission of a packet without waiting for either a negative acknowledgement (NAK) from the receiver or a timeout. In this case it is possible for the sender to transmit a packet multiple times so that more than one copy of a given packet may be correctly received at the decoder. We term this resulting rate penalty the *transport redundancy* introduced by the scheduling algorithm. This is different from most traditional ARQ approaches applied in video applications that only retransmit a packet upon the detection of a packet error or loss. This type of redundancy has not been explicitly studied in previous research. In this work, we investigate the impacts of both transport and source redundancy on the error control for a lossy packet network. From our experiments we make the following observations. First, regardless of whether source redundancy exists or not, a well-controlled transport redundancy through the Lagrangian optimized scheduling algorithm can improve the end-to-end performance in a delay-sensitive application. Second, in the absence of transport redundancy, source redundancy helps combat channel errors especially in high packet loss rate or under stringent delay constraints. Finally, these two types of redundancy can complement each other and achieve efficient video streaming even with very poor channel conditions, for example, at very high packet loss rates or relatively long RTT as compared to the end-to-end delay.

The paper is organized as follows. In Section II, we briefly review the basic RaDiO framework in [10]. Section III starts with a discussion of the proposed MDLC codec, and then describes a general DAHG source model that uses the MDLC as an example, the expected end-to-end distortion, and the analysis of its Taylor expansion. Section IV proposes the rate-distortion based scheduling algorithms based on the DAHG model, and describes the concept of transport redundancy. Simulation results are presented

in Section V. Conclusions and future work are discussed in Section VI.

II. Review of Basic RaDiO Framework

In this section we briefly review the rate-distortion optimized streaming framework of [10]. A compressed media stream is packetized into packets or data units. Here, we simply assume each data unit is put into one packet, and in the following discussion we do not differentiate between a data unit and a packet. The source dependencies between a group of data units are modelled as a directed acyclic graph (DAG), in which each vertex represents a data unit, and each directed edge from data unit i to data unit j indicates the decoding dependence of j on i, i.e., data unit j can only be decoded if i is received and decoded. Associated with each data unit l in the graph are three constant quantities: its size r_l in bytes, its time deadline t_l, i.e., the time by which it must arrive at the receiver to be useful for decoding, and its distortion value d_l, i.e., the amount by which the distortion of the decoded video will decrease if l is decoded on time at the receiver. The model implicitly assumes that when each data unit becomes decodable the total distortion is reduced by its distortion value.

The streaming system decides whether, when and how to transmit each data unit in a way that maximizes the playback quality at the decoder under the given network conditions and application requirements. This framework assumes that data units are transmitted at discrete intervals of time. At each transmission time, a data unit is chosen for transmission from those whose deadlines fall within a limited time window. Transmission decisions for such a group of data units at discrete times can be described by a transmission policy $\boldsymbol{\pi}$. For a group of L data units, $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_L]$, in which π_l is a binary vector indicating whether data unit l will be transmitted or not at each of the available transmission opportunities, unless there is an acknowledgement that l has been received. At each transmission time, the algorithm determines which data units to send by optimizing its transmission policy for the current transmission opportunity together with a complete plan for future transmission opportunities that will likely happen. The optimal policy $\boldsymbol{\pi}^*$ is the one that minimizes the expected Lagrangian cost function using a Lagrange multiplier λ,

$$J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi}), \tag{1}$$

where $D(\boldsymbol{\pi})$ is the expected end-to-end distortion and $R(\boldsymbol{\pi})$ is the expected transmission rate for a given $\boldsymbol{\pi}$. Based on the DAG model, $D(\boldsymbol{\pi})$ is given by

$$D(\boldsymbol{\pi}) = D_0 - \sum_l d_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})) \tag{2}$$

where D_0 is the distortion of the media stream if no packets are decoded, $\epsilon(\pi_l)$ is the packet loss probability of data unit l under policy π_l (strictly speaking, the probability that l is lost or does not arrive at the receiver on time), and $\prod_{l' \preceq l}(1-\epsilon(\pi_{l'}))$ is the probability that l is decodable. $l' \preceq l$ refers to the set of data units that must arrive at the receiver for l to be decoded. The given policy $\boldsymbol{\pi}$ also induces an expected number of transmission times, $\beta(\pi_l)$, for each data unit l, and $R(\boldsymbol{\pi}) = \sum_l r_l \beta(\pi_l)$.

An iterative descent algorithm was proposed in [10] to find $\boldsymbol{\pi}^*$. The algorithm starts with an initial policy, and then proceeds to minimize (1) iteratively until $J(\boldsymbol{\pi})$ converges. At each iteration step, (1) is minimized with respect to π_l while fixing the transmission policies of other data units, $\pi_{l'}$, $l' \neq l$. The optimization is done for different data units in a round-robin order. To optimize π_l, (1) can be rewritten as $J(\pi_l) = \epsilon(\pi_l) + \frac{\lambda r_l}{a_l}\beta(\pi_l)$, where a_l is the partial derivative of (2) with respect to $\epsilon(\pi_l)$, indicating the sensitivity (or importance) of receiving data unit l to the overall distortion. $\boldsymbol{\pi}$ is re-optimized at each transmission opportunity to take into account the feedback information and possible changes of the group of data units since the previous transmission opportunity.

III. Source Modelling for Redundant Representations

A. Example: MDLC

The MDLC system we proposed in [20] combines the bandwidth adaptation flexibility of LC with the reliability of MDC. An MDLC encoder first generates two base layer descriptions BL_1 and BL_2 using MDC approaches [8,9]. Then the base layer decoder module inside the MDLC encoder replicates three possible decoding scenarios at the receiver, i.e., both descriptions received or either one received. Based on each possible scenario of base layer reconstruction, an enhancement description is created by coding the difference between the original source and the base layer reconstructed signal. In general there are three enhancement layer descriptions: EL_0 when both BL_1 and BL_2 are received, EL_1 when only BL_1 is received, and EL_2 when only BL_2 is received.

In this paper, we use video redundancy coding [28,30] to create a MDC base layer, by partitioning a video sequence into two subsequences each of which mainly contains either odd or even frames. At the base layer, each subsequence is coded independently as an IPP sequence, where only the first frame (I-frame) of each group of pictures (GOP) is shared between both subsequences, as shown in Fig. 1(a). Coding efficiency is reduced because the motion-compensated prediction using a past frame farther apart is usually less efficient than using the immediately past frame. If both descriptions are received correctly, each bit stream is decoded independently to produce even and odd frames that are interleaved for the final base layer reconstruction. However, if only one description is received, the missed description can

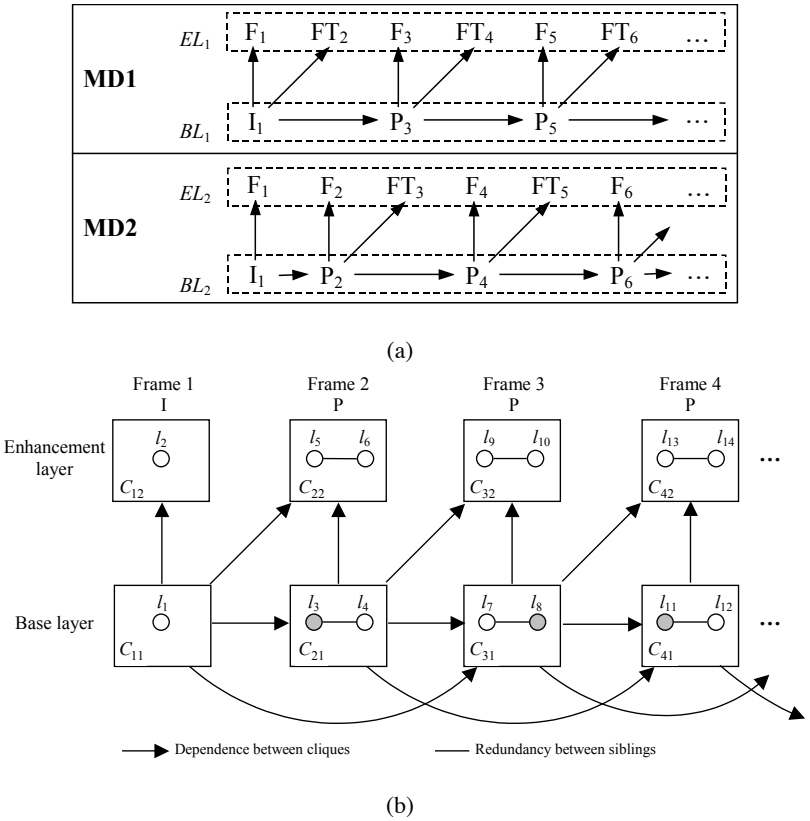


Fig. 1. MDLC scheme based on MPEG-4 FGST. (a) MDLC scheme. I: I-frame, P: P-frame, F: the enhancement layer generated by coding the residual between the original frame and its base layer reconstruction, FT: the enhancement layer generated by FGST using forward prediction from the base layer of its previous frame. The subscript of each label indicates the frame number. EL_0 is simply composed of F_i identical to either EL_1 or EL_2 based on the frame index. (b) The DAHG model of the above MDLC scheme. One of the two base layer nodes (filled with gray color), which has zero data size, is decoded as a copy or motion interpolation from the other description. We label each node sequentially as $l_1, l_2, \cdots$ starting from frame 1. Specifically, in frame 2, l_3, l_4, l_5 and l_6 correspond to BL_1, BL_2, EL_1 and EL_2, respectively.

be estimated by simply copying the closest adjacent frame in the correctly received description or using more complicated motion interpolation techniques by exploiting both past and future frames [29].

In order to construct a MDLC codec, we introduce additional fine granularity bit-rate scalability by generating enhancement layers upon base layer descriptions. For each subsequence, as shown in Fig. 1(a), we create enhancement layer descriptions with the MPEG-4 FGS temporal scalability (FGST) approach [19]. Each enhancement layer description codes the residual DCT coefficients between the original picture and a reference picture reconstructed from its corresponding base layer description in bit-plane coding. The reference picture is obtained in different ways depending on whether a frame is coded in the base layer description or not. Without loss of generality, consider the P-frame with an odd-

index i in Fig. 1(a). Its enhancement layer EL_1 (denoted by F_i) represents the residue between frame i and its BL_1 reconstruction, while its EL_2 (denoted FT_i) is generated using forward prediction from the BL_2 reconstruction of the previous frame $i-1$. FT_i also contains information of enhancement-layer motion vectors. At the decoder, depending on what base layer description is received, the enhancement layer can choose to decode either all (e.g., when both base layer descriptions are received) or a subset of the descriptions (when only one base layer description is received). The final enhancement layer quality is the one with the best quality achieved by all decodable descriptions.

Other extensions of this MDLC approach are not considered in this paper, as it is simply used as an example to demonstrate the efficiency of our scheduling algorithm for source codecs with redundancy. In fact, this particular approach has a number of practical advantages in addition to general features of MDLC. First, in addition to SNR quality, it provides temporal scalability that leads to a good reconstruction at half the original frame rate even when only one description is received. Second, it can be easily combined with multiple path transport to improve error resilience. Third, it is straightforward to expand the current approach to more than two descriptions by splitting the frames evenly into multiple independent subsequences and coding each enhancement layer description using the same FGST approach. Last, it has the flexibility to provide unbalanced base layer descriptions by using different quantization steps for each description, which is useful to cover a wide range of bit rates for bandwidth adaptation.

B. Directed Acyclic Hypergraph (DAHG)

When a video sequence is encoded into multiple redundant representations, source redundancy is introduced between two data units, where each of them can be decoded independently to create different representations of the same source unit. Such examples include BL_1 and BL_2 in MDLC, or data units that contain individual independent encodings of a frame with different quantization parameters. The key problem here is how to represent the redundancy between data units, and furthermore the possible availability of multiple decoding paths due to the redundancy.

To address this class of source coding formats, we introduce a new source model called Directed Acyclic HyperGraph (DAHG) to represent both dependency and redundancy relationships between different video data units. A DAHG is a generalization of a DAG $G = (V, E)$ where

1) Each vertex $C \in V$, rather than being a simple node, is composed of a set of nodes, each pair of which is connected by an undirected edge. We name this type of vertex a "clique", representing a collection of data units that produce multiple redundant representations of the same source coding unit, such as a frame or a SNR layer of a frame in scalable coding. Each node (or data unit) in a clique

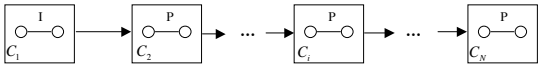


Fig. 2. Another example of DAHG to represent multiple independent encodings of a video sequence or error concealment.

represents one encoded version of this source unit, and an undirected edge connecting two nodes in the same clique indicates the redundancy between different encoded versions. A pair of nodes i and j are called *siblings*, and we write $i \sim j$.

2) An edge $(C_1, C_2) \in E$, directed from clique C_1 to clique C_2, is used to represent that decoding of C_2 is directly dependent on C_1. C_1 is said to be a *parent* of C_2, and C_2 is said to be a *child* of C_1. A path is a sequence of vertices such that from each of its vertices there is a directed edge to the next vertex in the sequence. If a path leads from C_1 to C_2, then C_1 is said to be an *ancestor* of C_2, and C_2 is said to be a *descendant* of C_1, written as $C_1 \prec C_2$ or $C_2 \succ C_1$. Each parent of C_2 is certainly an ancestor of C_2. On the other hand, C_1 being an ancestor but not a parent of C_2 indicates an indirect decoding dependence between C_1 and C_2. For example, this would be the case with last P-frame in a GOP (as C_2) depending on the first I-frame (as C_1) through the other intermediate P-frames.

Fig. 1(b) shows an example DAHG for the proposed MDLC scheme. Each frame i contains a base layer clique C_{i1} and an enhancement layer clique C_{i2}. Since the I-frame of each GOP is coded without redundancy, its base and enhancement layer cliques contain only one node each. Clique C_{i1} of each P frame i consists of two nodes representing BL_1 and BL_2, respectively. One of them is generated by copying (or using motion interpolation on) neighboring frames coded in the other description, such as l_3 of C_{21} in the figure. While this node does not require bits being sent it does produce a distortion reduction. Clique C_{i2} contains nodes EL_1 and EL_2. Directed edges connecting cliques represent either SNR dependence or temporal dependence. There are two directed edges entering C_{32}, including (C_{31}, C_{32}) for SNR dependence and (C_{21}, C_{32}) for temporal dependence. Thus, C_{31} and C_{21} are parents of C_{32}. C_{11} is an ancestor of C_{32} as a path $(C_{11}, C_{21}, C_{31}, C_{32})$ leads from C_{11} to C_{32}. Fig. 2 models multiple independent encodings of a video sequence, where the sequence is independently coded twice with different quantization steps using a typical non-scalable codec. Each clique contains two nodes to represent each encoded version. The same graph model can also be applied to a simple "copy previous frame" error concealment method, where one of the two nodes in each clique represents a duplicate copy of the previous frame as an approximation of the current frame.

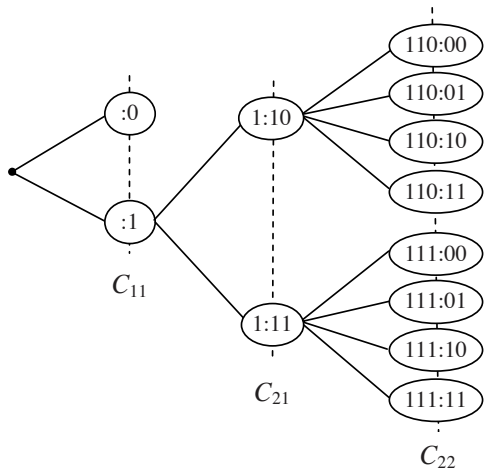
Assume that a clique contains N data units. Since each unit can be either received correctly or not

received (due to loss or because it is not transmitted in the first place), there are a total of 2^N possible states for the clique. A *clique state* is represented by a length-N binary string s, with each bit indicating the status of a data unit in the clique. Let b_l denote the corresponding bit location of data unit l in s; the b_lth bit of s is 1 (mathematically, $s[b_l] = 1$) if l arrives at the receiver on time and is 0 otherwise. b_l is set to 1 for those nodes that have a size of zero bits, since they are regarded as being always received[1]. Zero state of a clique is then defined as the state such that no data units are received, and all the other states that have at least one data unit received are called non-zero states. Note that a non-zero clique state does not necessarily mean that this clique is decodable. Decoding of a clique also depends on the states of its ancestor cliques, which will be discussed later. In addition, it is convenient to define $B_C^{(s)} = \{l | l \in C, s[b_l] = 1\}$ and $\bar{B}_C^{(s)} = \{l | l \in C, s[b_l] = 0\}$ to represent two different sets of data units in C based on their state s.

In a directed acyclic graph, a decoding path leading to a vertex can be constructed as an ordered list of its ancestors in the decoding order. In past works that code a video sequence into a single encoded version, such as single description coding, a vertex has only 1/0 states, i.e., either received or not. Thus each vertex node along a decoding path must be received in order for the current node to be decoded, and this forms a single decoding path. In contrast, in the case of source coding with redundancy, a clique can be decoded once all its ancestor cliques are received in a non-zero state. Moreover, each clique in the ordered ancestor list can take multiple non-zero states, with different state combinations resulting in possibly different decoded versions of the current clique. A *decoding path* leading to clique C is then defined as a particular combination of all C's ancestor clique states. Multiple decoding paths become possible as each ancestor may have multiple clique states. In order to use the same mathematical notation, we simply assume there is one virtual decoding path leading to those cliques that do not have parents. Fig. 3 shows the concept of multiple clique states and multiple decoding paths using the MDLC example.

In summary, DAHG is different from DAG mainly in two aspects: (1) multiple decoding paths in DAHG vs. single decoding path in DAG, and (2) multiple decodable clique states in DAHG vs. 0/1 state of the data unit (i.e., it is either decodable or not) in DAG. Estimating expected end-to-end distortion under a DAHG model will be discussed in detail in Section III-D.

[1]Examples of this type of nodes are given in Fig. 1(b) and 2. Essentially these nodes are created to separate the direct contribution of a packet to reducing distortion, which requires transmission, from its indirect contribution via interpolation or error concealment, which requires no additional transmission rate once the original packet has been received.



	C_{11}	C_{21}	C_{22}
Clique state s	$[l_1]$	$[l_3\ l_4]$	$[l_5\ l_6]$
Clique state set S	$\{[0],\ [1]\}$	$\{[10],\ [11]\}$	$\{[00],\ [01],\ [10],\ [11]\}$
Decoding path q		$[l_1]$	$[l_1\ l_3\ l_4]$
Decoding path set Q		$\{[1]\}$	$\{[110],\ [111]\}$

Fig. 3. Description of multiple clique states and multiple decoding paths using cliques C_{11}, C_{21} and C_{22} in Fig. 1 as an example. In frame 2, l_3 is decoded to be a direct copy of the reconstructed frame 1, and l_4 produces a reconstructed frame with better quality than l_3. Each circle in the figure is labelled by a combination of decoding path and clique state in the form "decoding path : clique sate". A decoding path is represented by a concatenation of each ancestor clique state. Nothing before the colon in C_{11} indicates that it has no parents and there is only a virtual decoding path leading to C_{11}.

C. Parameters Associated with DAHG

As in [10], each data unit l has a size r_l in bytes and a time deadline t_l by which it must arrive at the receiver to be useful for playback. However, the distortion reduction of a data unit in a DAHG model can take different values depending on the decoding path in which it is decoded. Let Q_C be the set of decoding paths leading to C. We can represent the distortion reduction of data unit l in the clique by a *distortion vector* $\mathbf{d}_l = [d_l^{(1)}, d_l^{(2)}, ..., d_l^{(q)}, ..., d_l^{(|Q_c|)}]$, where $d_l^{(q)}$ is the distortion reduction if l is decoded in the qth decoding path, and $|.|$ denotes the cardinality of the set. Setting $d_l^{(q)}$ to 0 will force the scheduler not to transmit data unit l given the qth decoding path. This can be used to eliminate certain undesirable clique state combinations, and thus reduce the number of effective decoding paths in a DAHG.

Though each of the data units in clique C can produce a certain distortion reduction, the total distortion reduction when more than one data unit is received correctly is usually less than the sum of their respective distortion reductions. Let S_C be the set of all clique states in C. We introduce a *redundancy matrix* $\mathbf{I}_C = [I_C^{(s,q)}]$ of dimension $|S_C| \times |\ Q_C|$, to represent the redundancy between different data units inside the same clique C. The redundancy of C, when it is in state s and decoded in the qth decoding path, is stored as an entry in row s and column q of the redundancy matrix. $I_C^{(s,q)}$ is defined as

$$I_C^{(s,q)} = \sum_{l \in B_C^{(s)}} d_l^{(q)} - d_C^{(s,q)}, \tag{3}$$

where $d_C^{(s,q)}$ is the distortion reduction of C if it is decoded in state s and the qth decoding path. An important property of this model is that, as the DAG model, the distortion reduction is still additive at the clique level; however, the amount by which the distortion decreases when a node is decoded depends not only on the state of its ancestor cliques but also on whether its siblings in the same clique are decodable. Fig. I lists the distortion vectors and redundancy matrices of C_{21} and C_{22} in Fig. 3.

D. Expected End-to-End Distortion

Suppose we already have a DAHG model to represent a group of L data units, with each data unit being packetized into one packet. We now estimate the expected end-to-end distortion of this group of packets (GOPkt) when given a vector of packet loss probability (PLP) providing a loss probability for each packet in the group. Recall that a packet is considered lost if it is either lost or arrives at the decoder too late to be played. We define the "*transmission state*" as the PLP vector which accounts for the transmission schedules and the channel conditions. Let ϵ_l be the PLP of packet $l \in \{1, ..., L\}$ and let $\boldsymbol{\epsilon} = [\epsilon_1, ..., \epsilon_L]$ be the real-time transmission state. Computation of expected distortion in a DAHG for a given $\boldsymbol{\epsilon}$ differs from that in [10] by introducing two new concepts, multiple decoding paths and multiple decodable clique states.

To help us write an expression of the expected distortion, we first derive some related probabilities. The probability of occurrence of clique state s is given by

$$p_C^{(s)} = \prod_{l \in B_C^{(s)}} (1 - \epsilon_l) \prod_{l' \in \bar{B}_C^{(s)}} \epsilon_{l'} \tag{4}$$

Recall that a decoding path leading to clique C is defined by a particular combination of the clique states of all its ancestors. Thus the probability of occurrence of decoding path q can be written in terms of the probabilities of those clique states as

$$p_C^{(q)} = \prod_{C' \prec C, s_{C'} \in q} p_C^{(s_{C'})} = \prod_{l \in A_C^{(q)}} (1 - \epsilon_l) \prod_{l' \in \bar{A}_C^{(q)}} \epsilon_{l'} \tag{5}$$

where $A_C^{(q)} = \bigcup_{C' \prec C, s_{C'} \in q} B_C^{(s_{C'})}$, and $\bar{A}_C^{(q)} = \bigcup_{C' \prec C, s_{C'} \in q} \bar{B}_C^{(s_{C'})}$. We can now write the expected distortion as a function of the transmission state

$$D(\boldsymbol{\epsilon}) = D_0 - \sum_C \sum_{q \in Q_C} p_C^{(q)} [\sum_{s \in S_C} p_C^{(s)} d_C^{(s,q)}] \tag{6}$$

where D_0 is the distortion of the GOPkt if no packets are decoded, $d_C^{(s,q)} = \sum_{l \in B_C^{(s)}} d_l^{(q)} - I_C^{(s,q)}$ directly derived from (3), and $p_C^{(s)}$ and $p_C^{(q)}$ are defined in (4) and (5), respectively.

Both transmitting and receiving a packet cause a state transition from a state $\boldsymbol{\epsilon_1}$ to another state $\boldsymbol{\epsilon_2}$. The Taylor expansion of D in terms of $\boldsymbol{\epsilon}$ reveals different characteristics of state transitions for different coding scenarios. The distortion reduction when receiving a packet in a multiple-decoding-path scenario depends on more factors than that in a single-decoding-path scenario, because the redundancy between packets plays an important role. Thus, in this case, an optimal scheduling algorithm should be designed to take into account both dependency and redundancy such that the end-to-end distortion is minimized at the decoder.

Taylor expansion of (6) at the current state $\tilde{\boldsymbol{\epsilon}}$ is given by

$$\begin{aligned} D(\boldsymbol{\epsilon}) &= \sum_{k=0}^{\infty} \Big[\frac{1}{k!} (\Delta\boldsymbol{\epsilon} \cdot \nabla_{\boldsymbol{\epsilon'}})^k D(\boldsymbol{\epsilon'}) \Big]_{\boldsymbol{\epsilon'}=\tilde{\boldsymbol{\epsilon}}} \\ &= D(\tilde{\boldsymbol{\epsilon}}) + \sum_i a_i \Delta\epsilon_i + \sum_{i,j} a_{ij} \Delta\epsilon_i \Delta\epsilon_j + \dots \end{aligned} \tag{7}$$

where $\Delta\epsilon_i = \epsilon_i - \tilde{\epsilon}_i$, $a_i = \frac{\partial D}{\partial \epsilon_i}$, and $a_{ij} = \frac{\partial^2 D}{\partial \epsilon_i \partial \epsilon_j}$. Note that (7) only contains linear terms, since $\forall\ 1 \le j \le k,$ if $\exists\ m_j \ge 2,$ then $\frac{\partial^n D}{\partial \epsilon_{i_1}^{m_1}, \cdots, \partial \epsilon_{i_k}^{m_k}} = 0$, derived directly from (6). a_i indicates the importance of packet i in terms of its contribution to the overall distortion reduction given the current transmission state. As receiving a packet will not increase the overall distortion for any coding application, $a_i \ge 0$ for any i. The second or higher-order derivatives take effect when there is more than one packet whose PLP has changed from a reference state. For example, $\frac{\partial^2 D}{\partial \epsilon_i \partial \epsilon_j}$ shows that a future change of ϵ_j, as packet j is transmitted or its ACK/NAK is received, may affect the importance of transmitting packet i at the current time. To see this, we approximate a_i by its first-order Taylor expansion at $\tilde{\boldsymbol{\epsilon}}$, $a_i(\boldsymbol{\epsilon}) \approx a_i(\tilde{\boldsymbol{\epsilon}}) + \sum_j a_{ij}(\epsilon_j - \tilde{\epsilon}_j)$. Assume that packet j will be transmitted or will arrive at the receiver when the state transits from $\tilde{\boldsymbol{\epsilon}}$ to $\boldsymbol{\epsilon}$, then $\epsilon_j < \tilde{\epsilon}_j$. In this case, a_{ij} will lead to a change in a_i as follows: when $a_{ij} < 0$, a_i increases and vice versa. In other words, the transmission or arrival of packet j may increase or reduce the current importance of packet i depending on the sign of a_{ij}.

Now we look at the properties of a_i and a_{ij} for both single decoding path and multiple decoding paths. First consider the single decoding path case. We derive its partial derivatives from (2),

$$\frac{\partial D}{\partial \epsilon_i} = \sum_{l \succeq i} d_l \prod_{l' \preceq l, l' \neq i} (1 - \epsilon_{l'}) \tag{8}$$

$$\frac{\partial^2 D}{\partial \epsilon_i \partial \epsilon_j} = -\sum_{l \succeq i,j} d_l \prod_{l' \preceq l, l' \neq i,j} (1 - \epsilon_{l'}) \tag{9}$$

The right hand side of (8) can be written as the sum of two terms f_1 and f_2, where $f_1 = d_i \prod_{l' \prec i} (1 - \epsilon_{l'})$ corresponds to the original distortion of packet i weighted by the probability of receiving all its ancestors,

and $f_2 = \sum_{l \succ i} d_l \prod_{l' \preceq l, l' \neq i} (1 - \epsilon_{l'})$ indicates the importance of packet i to its descendant packets. From (9), we can conclude $\frac{\partial^2 D}{\partial \epsilon_i \partial \epsilon_j} \leq 0$ for any i and j, since $\epsilon_{l'} \leq 1$ for any l'.

When multiple decoding paths are possible, we derive its first-order derivative from (6) as

$$\frac{\partial D}{\partial \epsilon_i} = f_1 + f_2 + f_3 + f_4 \tag{10}$$

with

$$f_1 = \sum_{q \in Q_{C_i}} p_{C_i}^{(q)} \Big[\sum_{s \in S_{C_i}, i \in B_{C_i}^{(s)}} d_{C_i}^{(s,q)} \prod_{l \in B_{C_i}^{(s)}, l \neq i} (1 - \epsilon_l) \prod_{l \in \bar{B}_{C_i}^{(s)}} \epsilon_l \Big]$$

$$f_2 = - \sum_{q \in Q_{C_i}} p_{C_i}^{(q)} \Big[\sum_{s \in S_{C_i}, i \in \bar{B}_{C_i}^{(s)}} d_{C_i}^{(s,q)} \prod_{l \in B_{C_i}^{(s)}} (1 - \epsilon_l) \prod_{l \in \bar{B}_{C_i}^{(s)}, l \neq i} \epsilon_l \Big]$$

$$f_3 = \sum_{C \succ C_i} \sum_{q \in Q_C, i \in A_C^{(q)}} \Big[\prod_{l \in A_C^{(q)}, l \neq i} (1 - \epsilon_l) \prod_{l \in \bar{A}_C^{(q)}} \epsilon_l \Big] \cdot d_C^{(q)}$$

$$f_4 = - \sum_{C \succ C_i} \sum_{q \in Q_C, i \in \bar{A}_C^{(q)}} \Big[\prod_{l \in A_C^{(q)}} (1 - \epsilon_l) \prod_{l \in \bar{A}_C^{(q)}, l \neq i} \epsilon_l \Big] \cdot d_C^{(q)}$$

where C_i represents the clique that contains packet i, and $d_C^{(q)} = \sum_{s \in S_C} p_C^{(s)} d_C^{(s,q)}$ representing the average distortion reduction of C when it is decoded in the qth decoding path. f_1 indicates the packet importance due to its own distortion reduction; f_2 represents redundancy when both i and its sibling packets are received; f_3 shows the distortion reduction achieved by the descendant cliques of C_i in the decoding paths that require i to be received; and f_4 represents the impact of receiving i on the descendant cliques of C_i in the remaining decoding paths which do not require i to be received. The signs of these terms indicate whether it is desirable to transmit i or not when different packets have been received at the decoder in the past, as a positive (or negative) term will increase (or decrease) the value of $\frac{\partial D}{\partial \epsilon_i}$.

We now give a concrete example to illustrate how to calculate the expected distortion for the MDLC scheme in Fig. 1 and the properties of its partial derivatives in the case of multiple decoding paths. Here we only consider cliques C_{11}, C_{21} and C_{22}. Let ϵ_i denote the PLP of data unit l_i in Fig. 1(b). Since l_3 is a direct copy of l_1 without the need to send any bits, $\epsilon_3 = 0$. We use the notation of Fig. I for distortion related parameters.

(1) *Calculation of expected distortion:* The expected distortion D is given by

$$D = D_0 - \Delta D_{C_{11}} - \Delta D_{C_{21}} - \Delta D_{C_{22}} \tag{11}$$

where $\Delta D_{C_{11}} = (1-\epsilon_1) d_1, \Delta D_{C_{21}} = (1-\epsilon_1) \left[\epsilon_4 \;\; 1-\epsilon_4\right] \begin{bmatrix} d_3 \\ d_4 \end{bmatrix}, \Delta D_{C_{22}} = (1-\epsilon_1) \left[\epsilon_4 \;\; 1-\epsilon_4\right] \boldsymbol{d}_{C_{22}} \boldsymbol{p}_{C_{22}}$

TABLE I

DISTORTION VECTORS AND REDUNDANCY MATRICES OF C_{21} (TOP) AND C_{22} (BOTTOM) IN FIG. 3. LET d_i DENOTE THE DISTORTION REDUCTION OF l_i WHEN l_i HAS ONLY ONE DECODING PATH, AND LET $d_i^{(j)}$ BE THE DISTORTION REDUCTION OF l_i IN THE jTH DECODING PATH WHEN THERE ARE MULTIPLE DECODING PATHS. $\mathbf{I}_C[11]$ IS THE REDUNDANCY OF CLIQUE C AT STATE $s = [11]$. $\min(a, b)$ IS THE MINIMUM BETWEEN a AND b.

Decoding path	Distortion Vector		Redundancy
	$\mathbf{d}(l_3)$	$\mathbf{d}(l_4)$	$\mathbf{I}_{C_{21}}[11]$
[1]	d_3	d_4	$\min(d_3, d_4) = d_3$

Decoding path	Distortion Vector		Redundancy
	$\mathbf{d}(l_5)$	$\mathbf{d}(l_6)$	$\mathbf{I}_{C_{22}}[11]$
[110]	$d_5^{(1)}$	0	0
[111]	$d_5^{(2)}$	$d_6^{(2)}$	$\min(d_5^{(2)}, d_6^{(2)})$

with $\boldsymbol{d}_{C_{22}} = \begin{bmatrix} 0 & d_5^{(1)} & d_5^{(1)} \\ d_6^{(2)} & d_5^{(2)} & \max(d_5^{(2)}, d_6^{(2)}) \end{bmatrix}$ and $\boldsymbol{p}_{C_{22}} = \begin{bmatrix} \epsilon_5(1-\epsilon_6) \\ (1-\epsilon_5)\epsilon_6 \\ (1-\epsilon_5)(1-\epsilon_6) \end{bmatrix}$. Each entry of $\boldsymbol{d}_{C_{22}}$ at row q and column s gives the distortion reduction of C_{22} along the qth decoding path at a non-zero clique state s. The sth element of $\boldsymbol{p}_{C_{22}}$ is the probability of occurrence of C_{22} at state s.

(2) *First-order partial derivatives:* Take l_4 as an example to show the importance of $\frac{\partial D}{\partial \epsilon}$ to the system's behavior. Written as (10), $\frac{\partial D}{\partial \epsilon_4}$ is derived from (11) with $f_1 = (1-\epsilon_1)d_4$, $f_2 = -(1-\epsilon_1)d_3$, $f_3 = (1-\epsilon_1)(\boldsymbol{d}_{C_{22}}(2) \times \boldsymbol{p}_{C_{22}})$, and $f_4 = -(1-\epsilon_1)(\boldsymbol{d}_{C_{22}}(1) \times \boldsymbol{p}_{C_{22}})$, where $\boldsymbol{d}_{C_{22}}(q)$ $(q = 1, 2)$ is the qth row of $\boldsymbol{d}_{C_{22}}$. From the sign of the above terms, we can see that transmission of l_4 is more favorable when f_1 and f_3 are significant, and less favorable when f_2 and f_4 become dominant.

(3) *Second-order partial derivatives:* Assuming now l_1 and l_4 have been transmitted but without receiving acknowledgements yet, the current state $\tilde{\boldsymbol{\epsilon}} = [\epsilon_1, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6] = [\epsilon_1, 0, \epsilon_4, 1, 1]$, $0 \le \epsilon_1, \epsilon_4 \le 1$. Consider the following second-order derivatives at $\tilde{\boldsymbol{\epsilon}}$,

- $\frac{\partial^2 D}{\partial \epsilon_4 \partial \epsilon_6} = -(1-\epsilon_1)d_6^{(2)} \le 0$, since l_6 is dependent on l_4 for decoding;
- $\frac{\partial^2 D}{\partial \epsilon_5 \partial \epsilon_6} = (1-\epsilon_1)(1-\epsilon_4)I_{C_{22}} \ge 0$, since l_5 and l_6 have redundancy with each other.

Though it is complicated to derive a general equation of $\frac{\partial^2 D}{\partial \epsilon_i \partial \epsilon_j}$ from (6), we can see from the above example that, in the case of multiple decoding paths, $\frac{\partial^2 D}{\partial \epsilon_i \partial \epsilon_j}$ can be either non-negative or non-positive. In contrast, $\frac{\partial D}{\partial \epsilon_i \partial \epsilon_j} \le 0$ for single decoding path. This shows that, in the case of single decoding path, the arrival of one packet at the receiver can increase, or at least not reduce the importance of the other

packets, in terms of distortion reduction. However, when there are multiple decoding paths, due to the redundancy between packets which affects the high-order terms, the future transmission of packets may decrease the current importance value of a packet that contains redundant information.

E. Complexity Analysis

The computation of D and its partial derivatives involved in the scheduling algorithms includes two main parts: i) generation of rate-distortion (RD) data for all data units, in particular, the distortion vector $\mathbf{d}_l$ and redundancy matrix $\mathbf{I}_C$ and ii) calculation of D in (6) and $\frac{\partial D}{\partial \epsilon_i}$ in (10) at runtime given the RD data. Complexity cost is dominated by the generation of RD data (which may require to actually code the source with different decoding paths) and the overall computational complexity is critically dependent on the number of decoding paths.

In general a complete set of decoding paths leading to C contains all the combinations of C's ancestor clique states. Thus, theoretically, the number of decoding paths may increase exponentially in the number of ancestor cliques preceding C. However, in practical pre-encoded applications, for given a decoder implementation it is possible to simplify the source modelling so that several decoding paths can be pruned and merged, reducing the number of effective decoding paths and thus the complexity. There are two main mechanisms for pruning: i) paths that lead to poor quality will be ignored by the decoder and thus pruned from the source model; and ii) many decoding paths producing the same reconstruction can be merged into one decoding path. In addition, even when a decoder supports certain decoding paths, a source model for the purpose of scheduling can also choose to discard some of these paths in order to reduce the computation complexity, at the penalty of some performance loss. Refer to the example in Fig. 4, which is used for our experiments. Here, we only show the base-layer cliques for simplicity. In this example, we assume that the decoder reconstructs the missing base-layer frames by simply copying the past frame from the other description. However, this reconstruction has relatively poor quality and thus will not be used as a reference for future frames. In other words, once a frame is missing, all the subsequent frames in the same description will not be decoded. Thus, based on this decoder behavior, the path $l_1\bar{l_4}\bar{l_7}$ can be pruned because future frames in both descriptions will not be decoded whether they are received by the decoder or not. The paths $l_1\bar{l_4}l_7\bar{l_{12}}$, $l_1\bar{l_4}l_7l_{12}$, and $l_1l_4l_7\bar{l_{12}}$ are merged as they produce the same prediction for future frames (only description of $MD1$ is decodable).

Note that these pruning decisions can be made when designing the system, by observing the behavior of the codec on a small amount of training data. Thus, it is not necessary to generate RD data for a complete dataset in order to decide on pruning/merging strategies. For example, cross-description decoding of EL_2

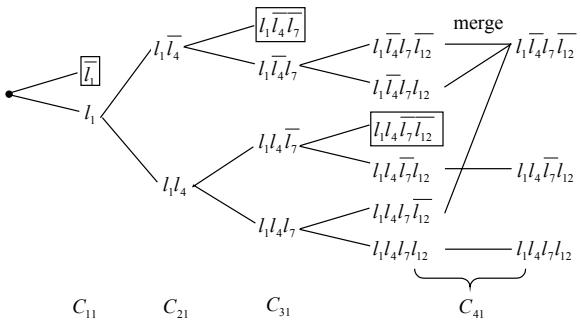


Fig. 4. Prune decoding paths of cliques C_{11}, C_{21}, C_{31} and C_{41} in Fig. 1. We use the label l_1 to indicate that node l_1 is received, and $\bar{l_1}$ when l_1 is not received. The other nodes follow the same notation. Those nodes that have a size of zero bits are not shown in the figure as they are regarded as being always received. Paths pruned at each stage are shown with a text that is enclosed with a frame box. In this example, the number of decoding paths passing through C_{41} is cut down from 16 (exhaustive) to only 3.

based on BL_1 or EL_1 based on BL_2 can typically be ignored since the information added by the cross enhancement layer is known to be very small. Similarly, if for a given codec it is known that two decoding paths tend to produce a similar reconstruction for typical content, then they can be merged a priori. Thus, after evaluation of RD behavior of encoder/decoder on a reduced training set, both collection of RD data and estimation of D and its derivatives can be performed on the resulting simplified source model with a much reduced complexity.

IV. Scheduling Algorithms with DAHG

A. System Architecture

In an end-to-end video transmission system, each video frame is encoded, transmitted and decoded in real-time within some acceptable delay. The input video is compressed into multiple redundant representations, e.g., using MDLC. For a packet-switched network, these streams are packetized and then fed into the transmission buffer. At each transmission time t, we make a selection decision only among packets whose playback deadlines fall within a time-varying transmission window $[lag(t), lead(t)]$. The time window will advance with t, and thus each packet has a limited number of transmission opportunities. $lag(t)$ is defined such that any packet whose playback deadline is earlier than $lag(t)$ could not arrive at the receiver on time if it were transmitted at t. $lead(t)$ implies the earliest time that a packet is eligible for transmission. [10] has proposed a number of ways to set $lead(t)$ by considering the receiver buffer implementation and application playback delay. Here we assume the end-to-end delay for each frame will

Algorithm 1 LAGRANGIAN(t, λ, $\boldsymbol{\pi}_{t-1}$)

1: $n = 0$: initialize $\pi_l = \{\pi_{l,t-1}, 0, \ldots, 0\}$ for each packet l, and calculate $\epsilon_l = \epsilon(\pi_l)$, $\beta_l = \beta(\pi_l)$, $D = D_0 - \sum_C \sum_{q\in Q_C} p_C^{(q)} [\sum_{s\in S_C} p_C^{(s)} d_C^{(s,q)}]$, $R = \sum_l r_l \beta_l$, $J = D + \lambda R$

2: **repeat**

3: $n = n + 1$

4: select packet l to optimize at step n in a round-robin order

5: $a_l = \frac{\partial D}{\partial \epsilon_l}$ obtained from (10)

6: $\pi_l^* = \text{argmin}_{\pi_l}\, \epsilon(\pi_l) + \frac{\lambda r_l}{a_l} \beta(\pi_l)$

7: $\epsilon_l = \epsilon(\pi_l^*)$, $\beta_l = \beta(\pi_l^*)$, $D = D_0 - \sum_C \sum_{q\in Q_C} p_C^{(q)} [\sum_{s\in S_C} p_C^{(s)} d_C^{(s,q)}]$, $R = \sum_l r_l \beta_l$, $J = D + \lambda R$

8: **until** $\left| J^{(n)} - J^{(n-1)} \right| <$ Threshold

9: **return** $\boldsymbol{\pi}^* = [\pi_1^*, \cdots, \pi_L^*]$

be constant and equal to the initial playback delay w. Thus, $lead(t) = lag(t) + w$. Each transmission at time t is subjected to a constraint of the admissible channel rates during this time interval. The receiver sends an ACK back to the sender as soon as it receives a packet. With the feedback information, the sender can estimate the channel conditions such as packet loss rate and round-trip time (RTT).

In our research, we simply model the network as an i.i.d. packet erasure channel with a fixed RTT, i.e., a packet sent at t is lost with probability ϵ independent of t. By time $t+$RTT, the sender will receive an ACK if the packet is received at the decoder; otherwise the packet is considered lost or corrupted. We also assume that the back channel is error-free. Thus, given the transmission policy that there are n transmission times of packet l in the last RTT, the expected PLP of packet l at time t is given by

$$\epsilon_l = \begin{cases} 0, & \text{if ACK for packet } l \text{ is received by } t, \\ \epsilon^n, & \text{otherwise.} \end{cases} \tag{12}$$

More complicated network models, such as random network delay and lossy back channel, can be easily combined into our streaming architecture and scheduling algorithms. The major difference for various network models is how to estimate the expected packet loss probability in (12) given a transmission policy. This part has been carefully studied in [10]. In this paper, we focus on the design of an efficient scheduling algorithm by taking into account both dependency and redundancy between packets.

B. Optimization Problem Formulation

The goal of scheduling is to minimize the playback distortion for a streaming session, by adapting to the network conditions and application requirements. Though we work with a more general streaming

framework that allows multiple decoding paths, we can follow the same problem formulation as originally proposed in [10] for streaming applications with single decoding path. Suppose we wish to transmit a group of L packets whose playback deadlines fall in a limited time window, and the packets are transmitted at discrete time intervals evenly distributed in a time window with a maximum of N transmission opportunities. Let $\pi_l = [v_0, \cdots, v_{N-1}]$ be the transmission policy for packet l along the N transmission opportunities, where $v_i = 1$ indicates "send packet l" and $v_i = 0$ "do not send packet l" at the ith time interval. We are interested in finding an optimal transmission policy $\boldsymbol{\pi}^* = [\pi_1, \cdots, \pi_L]$ for this group of packets such that the expected end-to-end distortion is minimized subject to the data rate constraint, i.e.,

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}: R(\boldsymbol{\pi}) \leq R_b}{\operatorname{argmin}} \; D(\boldsymbol{\pi}). \tag{13}$$

Since the expected PLP ϵ_l for packet l is a function of its transmission policy π_l, the expected end-to-end distortion D also depends on $\boldsymbol{\pi}$. Note that we consider expected distortion because there is uncertainty about the actual decoded video quality; changes in channel bandwidth, packet loss rate, and so forth will affect the quality of received video.

C. Lagrangian Optimization Algorithm

As proposed in [10], the constrained optimization problem in (13) can be cast as an unconstrained optimization problem using a Lagrange multiplier λ,

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \, D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi}). \tag{14}$$

As mentioned in Section II, $R(\boldsymbol{\pi}) = \sum_l r_l \beta(\pi_l)$. However, in this case, $D(\boldsymbol{\pi})$ is given by (6) using the DAHG model instead of (2). Our proposed scheduling algorithm is composed of two components: (1) at each transmission time t, the iterative descent Lagrangian optimization algorithm proposed in [10] is used to update $\boldsymbol{\pi}^*$ for a given λ, by taking into account the source rate-distortion information, current channel condition, transmission history and receiver feedback; (2) a window-based rate-control algorithm is applied regularly (e.g., at each transmission time) to adjust λ such that the average output rate of the scheduler is matched to the channel bandwidth.

Algorithm 1 describes the Lagrangian optimization algorithm for coding applications with multiple decoding paths. The major difference from [10] is that, at each optimization step, we derive the expected distortion from a DAHG model instead of a DAG, as the DAHG can well represent both dependency and redundancy between packets. The input parameter $\boldsymbol{\pi}_{t-1}$ represents the optimal transmission policy of the group of packets determined at previous time $t-1$. Since all the future transmission plans following

Algorithm 2 WINDOW_BASED_RATE_CONTROL(t, R_b)

1: **if** $t = 0$ **then**
2: $R_b = 0$
3: **if** M new frames come in **then**
4: $R_b = R_b + M *$ channel bandwidth $*$ frame interval
5: use bi-section algorithm to find an appropriate λ with rate constraint R_b
6: call LAGRANGIAN(t, λ, π_{t-1})
7: $R_b = R_b - \sum_{l:\pi_l(t)=1} r_l$
8: **return** R_b

current time t will be re-optimized, the function LAGRANGIAN is only interested in the segment of $\boldsymbol{\pi}_{t-1}$ that stores the transmission history up to $t-1$. Let $\pi_{l,t-1}$ denote the lth component of this past segment in $\boldsymbol{\pi}_{t-1}$. We first initialize the current transmission policy π_l of each packet to be the one with no further transmissions. At each iteration step, the Lagrangian cost J is minimized with respect to π_l of a selected packet l while keeping the policies of all other packets fixed. Upon convergence, π_l of each packet is optimized for its complete window of transmission opportunities. Then the transmitter takes transmission actions at t, and the optimization procedure will be repeated at $t+1$.

Next, in order to approach the channel bandwidth limit, we propose a window-based rate control scheme. That is, at each time, λ is fixed for all packets in the transmission window. The rate budget R_b is increased when a new frame enters into the transmission window, and decreased when packets are sent out. At each transmission time, we apply the bisection algorithm to find an appropriate λ for R_b. This approach is different from those that fix λ for each group of frames or the whole session in that it can quickly respond to channel bandwidth changes and use the bandwidth in a more efficient way. The rate control algorithm is summarized in Algorithm 2.

D. Greedy Algorithm

Since only the current transmission action in $\boldsymbol{\pi}$ is used at any given time, instead of determining the complete transmission policy for each packet over all possible transmission opportunities (e.g., as used in the above Lagrangian optimization), we could choose to use a greedy approach by selecting the currently most important packet from the group of L candidate packets. Previous research work [13] has proposed similar solutions for single-decoding-path applications. Here, we derive the greedy algorithm for multiple-decoding-path codecs from the Taylor expansion of the expected distortion. Given the past

Algorithm 3 GREEDY

1: **for all** $1 \leq i \leq L$ **do**

2: $\quad c_i = \epsilon^{m_i}\epsilon_i\frac{a_i}{r_i}$

3: Find the largest c_i, say j (i.e. $c_j \geq c_i$ for any $i \neq j$)

4: **return** j

transmission history of packet i, let $\boldsymbol{\pi}_{i,0}$ be a transmission schedule such that packet i is not transmitted at the current time t and all future time steps, and let $\boldsymbol{\pi}_{i,1}$ be the same transmission schedule as $\boldsymbol{\pi}_{i,0}$ except that packet i will be transmitted at t. Sending packet i at t induces a state transition from $\boldsymbol{\epsilon}(\boldsymbol{\pi}_{i,0})$ to $\boldsymbol{\epsilon}(\boldsymbol{\pi}_{i,1})$, and thus leads to a distortion reduction by

$$\begin{aligned}\Delta D_i^{(t)} &= D(\boldsymbol{\epsilon}(\boldsymbol{\pi}_{i,0})) - D(\boldsymbol{\epsilon}(\boldsymbol{\pi}_{i,1})) \\ &= a_i(\epsilon_{i,0} - \epsilon_{i,1}) = a_i(1-\epsilon)\epsilon_{i,0} \end{aligned} \tag{15}$$

derived from (7), where $\epsilon_{i,0}$ and $\epsilon_{i,1}$ are the PLP of packet i given the schedule $\boldsymbol{\pi}_{i,0}$ or $\boldsymbol{\pi}_{i,1}$, respectively. In fact, $\epsilon_{i,0}$ is the expected PLP of packet i at t given its transmission history as calculated in (12), and we simplify the notation to ϵ_i. $\Delta D_i^{(t)}$ indicates the importance of sending packet i at the current time t when no further transmissions are considered. To favor packets with early playback deadlines, we introduce a multiplier ϵ^{m_i} in (15), where m_i is designed to approximate the number of possible retransmissions by $m_i = (t_i - t)/RTT$. That is because the future possible transmissions for packet i will decrease the importance of sending it at t. Strictly speaking, the number of possible retransmissions can be much larger than the given m_i for a system that allows retransmissions without waiting. Ignoring the constant term $(1-\epsilon)$, for comparing the importance of sending each packet at t and taking into account the packet size r_i, we have the metric

$$c_i = \epsilon^{m_i}\epsilon_i\frac{a_i}{r_i} \tag{16}$$

for each packet and select the one with the largest c_i to send. Note that a_i is calculated at the current state with the assumption that there are no future transmissions of other packets. Algorithm 3 summarizes the proposed greedy technique. At each transmission time, we choose the most important packets to send by running this algorithm iteratively until the channel rate allocated to this time interval is used up.

A main problem of the greedy algorithm is that it ignores the possibility of future transmissions of other packets. As Section III-D points out, for applications with multiple decoding paths, the future transmission of a packet may either increase or decrease the importance value of another packet depending on their

coding relation. Thus, in an optimal algorithm future transmission probabilities of packets would have increased impact, through the higher-order derivatives of the Taylor expansion, on the decision at the current transmission opportunity. Another problem may arise from possible future retransmissions of the packet itself, for which this algorithm introduces a multiplier on the importance metric to approximate this impact on the current decision. Other work has studied improved greedy scheduling algorithm to address these problems. Our previous MDLC work in [20] proposed a double time window control to intentionally introduce an extra waiting period for description 2 (MD2) such that it can only be transmitted relatively safely in a future time to avoid unnecessary redundancy with description 1 (MD1). This helps when the acknowledgements generated by early transmissions of MD1 are likely to arrive at the sender soon. [16] proposed to delay some packet scheduling decisions in order to avoid the penalty introduced by premature retransmissions. However, for a general coding scenario that provides multiple decoding paths, we have not achieved a systematic solution to address the possible future (re)transmissions for the packet itself and its related packets (through either dependency or redundancy). We are now working on a possible solution by taking into account the higher-order partial derivatives in Section III-D.

E. Transport Redundancy

Traditional ARQ approaches request retransmission only upon detection of lost or overly delayed packets. Thus the number of retransmissions is very limited for delay constrained real-time video communication. In comparison, our extended streaming framework, together with the one originally proposed in [10] for single decoding path, allows unlimited retransmission of a packet before the playback deadline in the sense that it can retransmit a packet without waiting for a timeout or NAK from the receiver. This approach essentially relieves the delay problem caused by retransmissions. However, it may introduce a rate penalty when both retransmitted and original packets are correctly received at the decoder. We call this "transport redundancy", since the client receives redundant information[2].

One possible variation of the proposed scheduling algorithms is to mimic the traditional ARQ systems by limiting the retransmission of a packet until the last transmission of this packet has not been acknowledged within a predefined timeout. Based on our system assumption with a fixed RTT, the timeout is simply defined to be equal to one RTT. In other systems where the network produces a random delay, the timeout can be set as the mean RTT plus some tolerance (e.g., three times the standard deviation of the

[2]If the original packet is not correctly received at the decoder, the duplicated packet contributes to the end-to-end distortion, and thus here we do not count it as a transport-redundant packet.

RTT, as is frequently used in ARQ systems). This is different from the original scheduling algorithms in that it completely or almost completely avoids the cost penalty due to the transport redundancy. However, if retransmission is controlled appropriately in the case where there is no waiting, the end-to-end performance can be improved without introducing longer delay. We will compare the performance of the scheduling algorithms with or without transport redundancy in Section V.

F. Complexity Analysis

The complexity of the Lagrangian optimization approach is on the order of $N_\lambda N_i L 2^N$ at each transmission time, where N_i is the number of iterations performed until the algorithm converges for a given λ, and N_λ is the number of iterations for the rate control algorithm to adjust λ to meet the rate limit. The time period to adjust λ could cross multiple transmission times in order to reduce the complexity. L is the number of packets available for transmission in the time window, and N is the number of transmission opportunities of a packet. The complexity of the Greedy approach is $O(L)$ since each packet only needs to be traversed once to choose the most important packet to send at a given time. Note that the limited retransmission variants of the proposed algorithms lead to decreases in complexity as the number of packets to be considered for transmission at each transmission opportunity decreases. Instead of considering L, we consider only those that have not been transmitted or have been transmitted in the distant past (e.g., one RTT ago) without acknowledgement. In addition, for Lagrangian optimization algorithm, the searching space of an optimal transmission policy for each packet is greatly reduced by the retransmission limitation.

V. Experimental Results

In this section, we examine the performance of the proposed streaming framework for video codecs with multiple decoding paths. The video sequences are coded using the proposed MDLC approach based on MPEG-4 FGST. Three standard test sequences are used: Akiyo (QCIF), Foreman (QCIF) and Mobile (CIF). The first 200 frames of each sequence are coded at 30f/s with a constant quantization step size. Each group of pictures (GOP) has 10 frames coded in IPP format. Specifically, at the base layer, 5 frames correspond to MD1 and 6 frames to MD2 in each GOP. Base layer reconstruction of missing frames is done by simply copying the past frame from the other description. Each base layer packet includes a complete frame. The enhancement layer of a frame in each description is coded bit-plane by bit-plane, with each bit-plane put into one packet. The performance is measured in terms of the average luminance peak signal-to-noise ratio (PSNR) in decibels of the decoded video frames at the receiver as a function

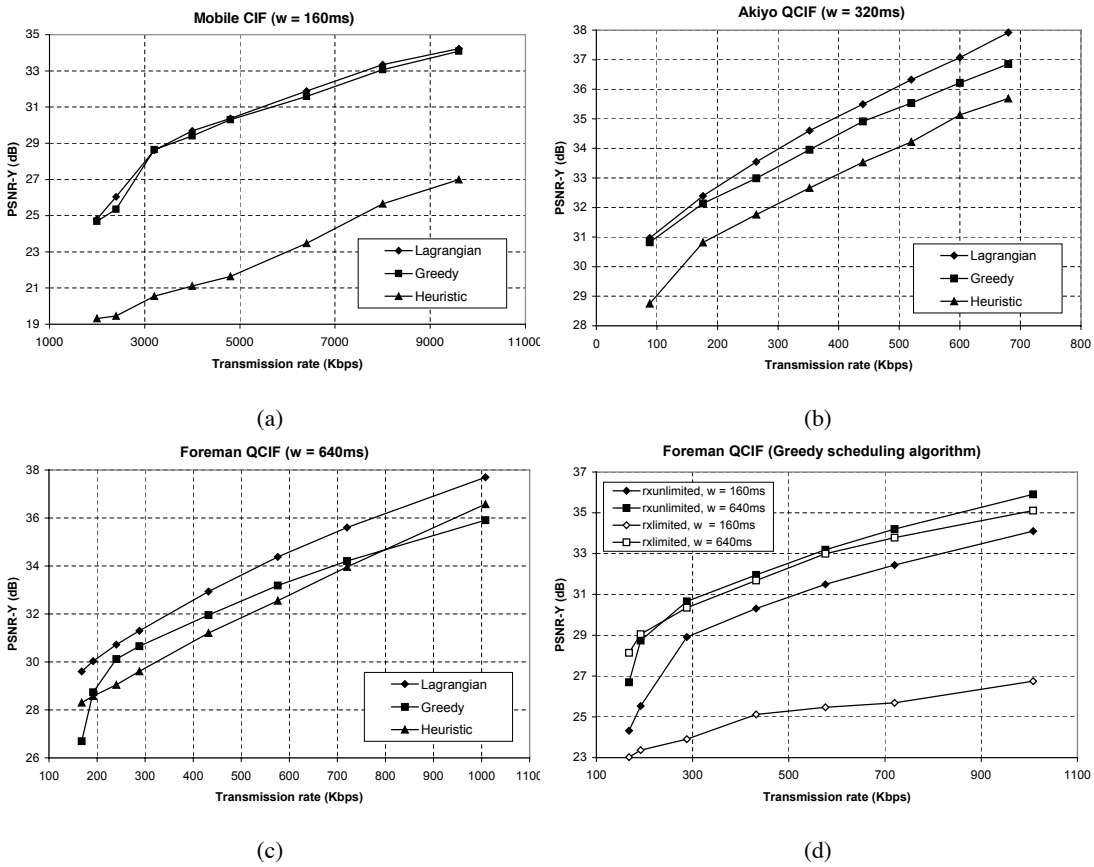


Fig. 5. Comparison between scheduling algorithms at PLR=0.15 for various playback delays. The base layer quantization parameters for Mobile, Akiyo, and Foreman are set to 12, 20, and 20, respectively.

of various system parameters, such as available channel bandwidth, packet loss rate (PLR), RTT and application playback delay (denoted by w). In all experiments, the channel RTT is set to 200 ms, and each packet has transmission opportunities every 80 ms. We performed 100 rounds for each experimental scenario and the results shown are the average of these rounds.

A. Comparison between Scheduling Algorithms

In addition to Lagrangian optimization and greedy algorithm, we also include in a comparison with a heuristic scheduling algorithm based on ARQ with prioritized transmission. In this algorithm, when the sender has not received the ACK of a packet after one RTT, it puts the packet back to the transmission queue for retransmission. The scheduler differentiates descriptions and layers by a predefined priority order. We choose the one that is observed in general to achieve better performance than the other orders. That is, $BL1$, $EL1$, $BL2$, $EL2$, in decreasing order of priority, which sends MD1 first and then MD2

for increased redundancy if additional rate is available (refer to Fig. 1(a)). Among packets in the same priority class, priority is given to those with earlier playback deadlines. Both Lagrangian optimization and the greedy algorithm allow retransmissions without waiting for a timeout.

Fig. 5(a)-(c) show the performance comparison between these systems when PLR $= 0.15$. First, the Lagrangian method provides substantial gains over the heuristic approach for the whole range of bandwidths under consideration at various playback delays. The performance gain is in the range of 1-7 dB and decreases as the playback delay increases. The heuristic approach prioritizes different descriptions and layers in a predefined order without exploiting rate-distortion information of source packets. Thus, the predefined order may lead to a mismatch between the added redundancy and that required by system conditions. For example, it does not introduce enough redundancy in the case of short delay as shown in Fig. 5(a), in that MD2 is not transmitted until all base and enhancement layers of MD1 have been sent. Furthermore, the number of retransmissions is restricted to be low due to the delay requirement. Therefore, the transmission of less significant enhancement layers of MD1 is more likely a waste of bandwidth due to the loss of its more significant layers. Second, the Lagrangian method outperforms greedy algorithm by up to 3 dB, and both algorithms achieve similar performance in the short playback delay case of Fig. 5(a). Greedy algorithm tends to introduce more redundancy in the system since it makes scheduling decisions without considering possible future packet transmissions. In some sense, greedy algorithm is not able to exploit a longer playback delay in a cost-efficient way. Finally, greedy algorithm performs better than the heuristic approach in most cases, since it exploits the knowledge of distortion impact of a packet loss on the reconstructed video quality. However, in the case of long playback delay, greedy algorithm performs poorly in some transmission rates for the reasons we just explained.

B. Redundancy's Role in Adaptive Streaming

We now describe a detailed performance analysis when two types of redundancy, namely source and transport redundancy, are used in the streaming system. We use the Lagrangian optimization algorithm as a default scheduling algorithm unless otherwise explicitly mentioned. For all experiments, we use LC as a representative single-decoding-path codec, and MDLC as an example of multiple-decoding-path codecs. In order to emphasize the impacts on the end-to-end distortion by different transmission policies and the adaptation flexibility introduced by source redundancy, we adjust the base layer quantization parameters (QP) so that MDLC and LC perform similarly in terms of coding efficiency at the encoder.With the choice of coding parameters, there is basically no loss in coding performance between an MDLC system and a LC system. In the experiments, for Foreman sequence, we set base layer QP to 24 for LC, and

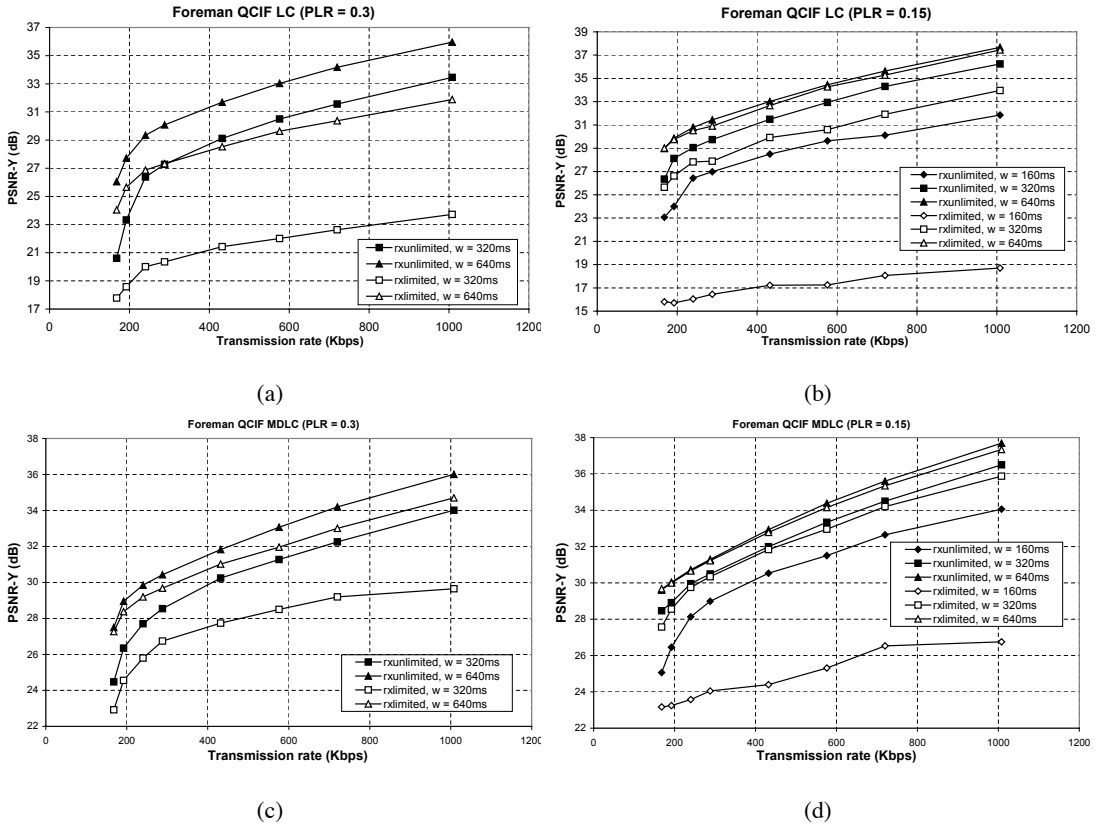


Fig. 6. The impact of transport redundancy on streaming performance when using Lagrangian optimization algorithm.

20 for both MD1 and MD2. For Mobile sequence, the same base layer QP of 12 is set for LC, MD1 and MD2. Three scenarios are considered in the experiments. First, we compare the performance with or without transport redundancy for each type of codec separately. Second, when transport redundancy is not available, we compare the performance of single-decoding-path and multiple-decoding-path codecs, i.e., LC and MDLC. Finally, we examine streaming performance when both source redundancy and transport redundancy are introduced in the streaming system.

1) Transport Redundancy: We first examine in Fig. 6 the performance of streaming Foreman, as a function of the available transmission rate and playback delay when LC and MDLC are used, respectively. Here, as discussed in Section IV-E, unlimited retransmission corresponds to the scheduling algorithms that allow retransmissions without waiting, while limited retransmission indicates the case where retransmissions have to wait till a timeout period. First, it can be seen that, for both source codecs, unlimited retransmission outperforms limited retransmission with a significant margin over the entire range of transmission rate. This is due to the fact that unlimited retransmission greatly increases the chance of

multiple retransmissions without incurring an unacceptable delay, and therefore the additional bandwidth can be efficiently used to retransmit the most important packets so as to improve the end quality at the receiver. Since the set of possible choices for $\boldsymbol{\pi}$ in (14) for limited retransmission is a subset of the corresponding set of unlimited retransmission, the Lagrangian optimization should ideally always achieve better performance when removing the retransmission restriction. Transport redundancy is well adjusted by Lagrangian optimization such that retransmissions are not wasted by exploiting the statistical knowledge of the channel and past transmission history. For example, it is observed that, if the playback delay is long enough, the scheduler chooses to wait for one RTT before initiating a new retransmission, so that retransmission only occurs if the sender does not receive the ACK. However, for an algorithm that does not take into account future packet transmissions, transport redundancy introduced by unlimited retransmissions may deteriorate the performance as shown in Fig. 5(d) with bandwidth below 200 Kbps when the greedy algorithm is used. Second, the performance gain of LC through transport redundancy tends to be more significant than that of MDLC in the same system setting. As seen in Fig. 6 (a) and (c) at $w = 320$ ms, the gain reaches up to 9 dB for LC and 4 dB for MDLC at high transmission rates. This is because source redundancy in MDLC provides a benefit similar to that of transport redundancy in terms of improving error robustness in a lossy packet network. Finally we observe that the performance difference between unlimited and limited retransmission is larger under poor channel conditions, such as high PLR and short playback delay. Thus limited retransmission may be appropriate as a lower complexity scheduling technique in the case of low PLR and long playback delay.

2) Source Redundancy without Transport Redundancy: We then compare in Fig. 7 the performance of LC and MDLC in the absence of transport redundancy, i.e., when Lagrangian algorithm is used with limited retransmissions. MDLC provides a significant gain over LC in the case of short playback delay and high PLR, where source redundancy introduced by multiple descriptions greatly improves error robustness. The performance gain achieves up to 8 dB for both Mobile and Foreman when $w = 160$ ms at PLR $= 0.15$. As w increases, the performance of LC improves as the number of possible retransmissions increases, and finally is close to that of MDLC at $w = 640$ ms. However, at very high PLR (e.g., PLR $= 0.3$), MDLC outperforms LC again with a gain of 1-3 dB when $w = 640$ ms. In very few cases, there is a penalty of up to 0.6 dB for MDLC over LC. This may be due to the possible local minimum involved in the Lagrangian optimization of MDLC. To summarize the results, source redundancy provides significant benefits on robust video communication especially in the case of high PLR and short playback delay, which is known to be a very difficult environment for video communication. When system conditions become favorable, source redundancy may not be necessary considering the additional

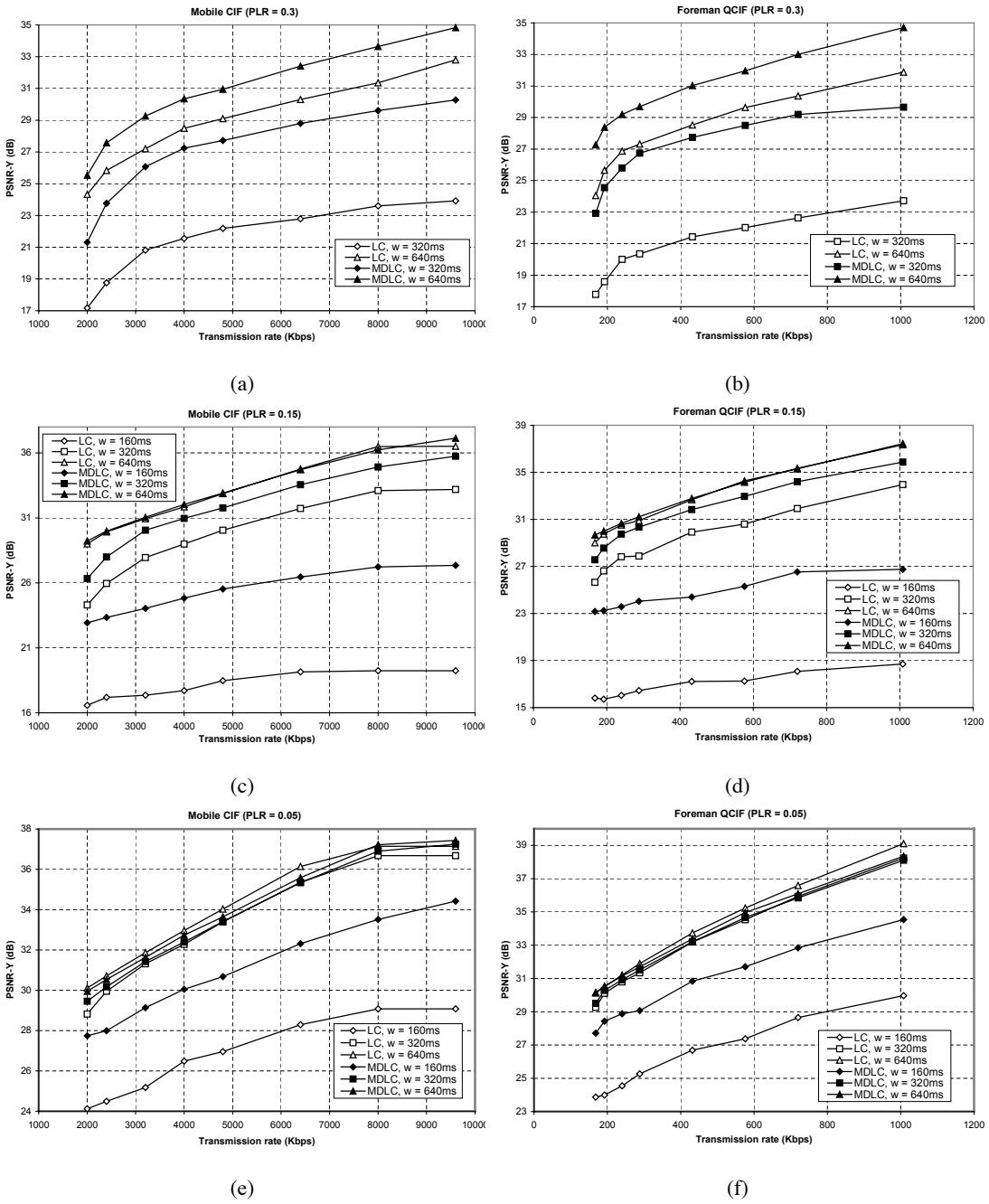


Fig. 7. Comparing LC and MDLC with limited retransmissions. The performance at $w = 160$ ms and PLR $= 0.3$ for both sequences is not included in the figure as the low PSNR achieved is out of acceptable range.

complexity it introduces.

3) Source Redundancy with Transport Redundancy: The final comparison is between LC and MDLC when transport redundancy is applied, i.e., when using the Lagrangian optimization algorithm with

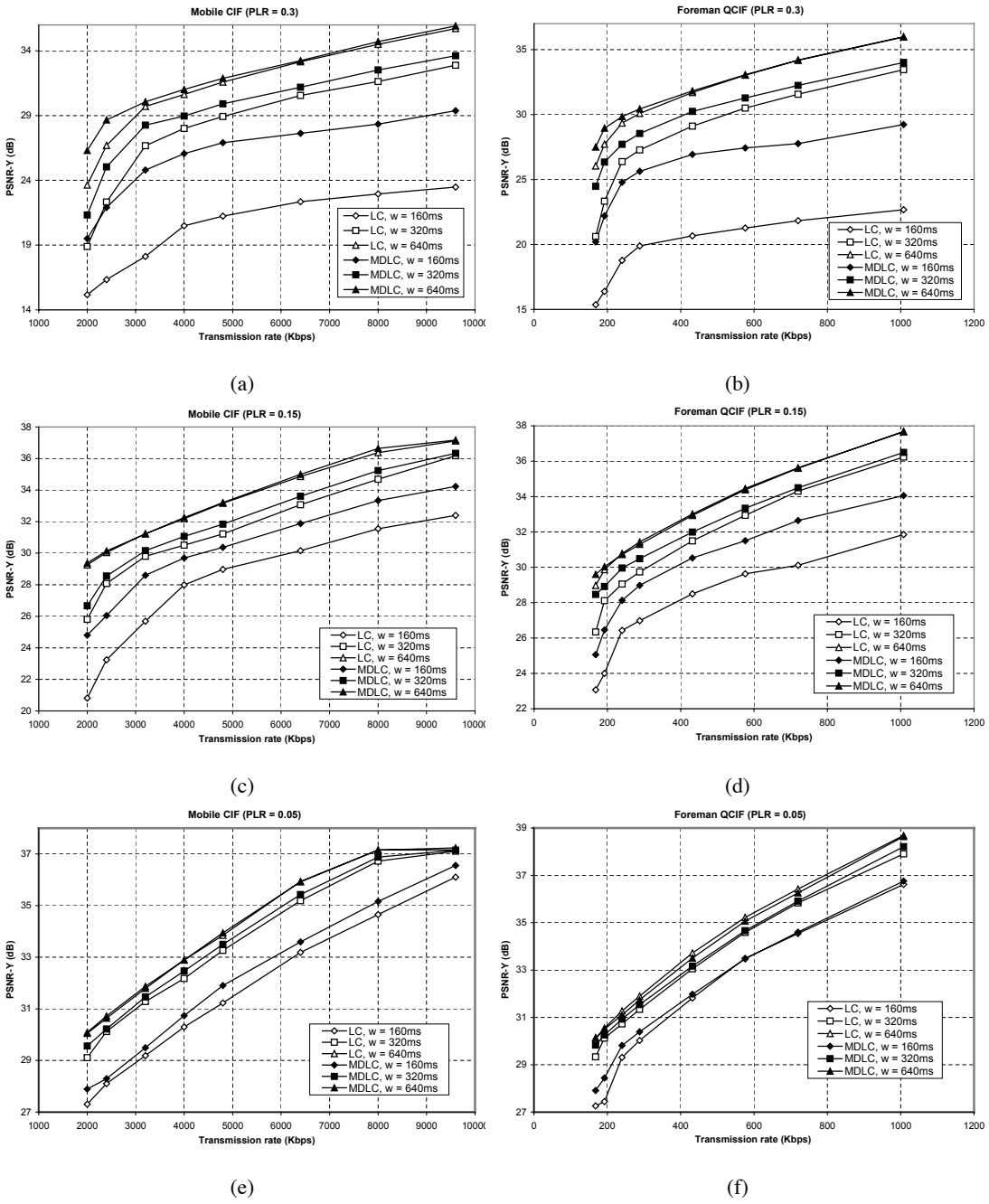


Fig. 8. Comparing LC and MDLC with unlimited retransmissions.

unlimited retransmissions. Fig. 8 shows the performance of streaming Mobile and Foreman under the same system settings as Fig. 7. Note that in this case the performance difference between MDLC and LC is not as large as in the previous case. This is expected as it is no longer necessary to wait for a timeout and thus packet retransmission becomes possible even in a delay-sensitive application, where

end-to-end delay is of the order of the RTT. The scheduling algorithm essentially provides unequal levels of transport redundancy to different packets based on their estimated importance, and thus overcomes the sensitivity of a LC system to transmission losses. But we can still observe a performance gain of up to 6 dB at $w = 160$ ms and PLR $= 0.3$. Similar to the second experiment, the performance gain varies as a function of channel conditions, and is larger for high PLR and low playback delay. Thus we can conclude that transport redundancy and source redundancy can both improve the end-to-end performance by enhancing the error robustness. Furthermore, the best performance is achieved when both types of redundancies are applied in the streaming system. Such a system can achieve efficient video streaming even under very poor channel conditions, such as for very high PLR or relatively long RTT compared to the playback delay.

VI. Conclusions and Future Work

In this paper we have extended recent work on rate-distortion based video scheduling to the general case where multiple decoding paths are possible. We proposed a new source model called Directed Acyclic Hypergraph (DAHG) to describe the decoding dependence and redundancy between different data units. Based on this model, we have proposed two rate-distortion based scheduling algorithms, i.e., the Lagrangian optimization and greedy algorithm. Experimental results demonstrate the performance improvement by exploiting coding relation and rate-distortion information of data units in the scheduling algorithms. The results show that our proposed system with both source and transport redundancy can provide very robust and efficient real-time video communication over lossy packet networks.

References

[1] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE J. Selected Areas in Communications*, vol. 17, pp. 1–18, May 1999.

[2] R. Zhang, S. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE J. Selected Areas in Communications*, vol. 18, pp. 966–976, June 2000.

[3] ITU-T Recommendation H.263 version 2 (H.263+), *Video coding for low bitrate communication*. Feb. 1998.

[4] B. Girod and N. Farber, "Feedback-based error control for mobile video transmission," *Proceedings of the IEEE*, vol. 87, pp. 1707–1723, Oct. 1999.

[5] Y. J. Liang and B. Girod, "Prescient r-d optimized packet dependency management for low-latency video streaming," in *Proc. Int'l Conf. Image Processing*, Sept. 2003.

[6] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Information Theory*, vol. 42, pp. 1737–1744, Nov. 1996.

[7] M. Podolsky, S. McCanne, and M. Vetterli, "Soft ARQ for layered streaming media," *The Journal of VLSI Signal Processing*, vol. 27, pp. 81–97, Feb. 2001.

[8] V. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, pp. 74–93, Sept. 2001.

[9] Y. Wang, A. Reibman, and S. Lin, "Multiple description coding for video delivery," *Proceedings of the IEEE*, vol. 93, pp. 57–70, Jan. 2005.

[10] P. Chou and Z. Miao, "Rate-distortion optimized sender-driven streaming over best-effort networks," in *Proc. Workshop on Multimedia Signal Processing*, vol. 1, pp. 587–592, Oct. 2001.

[11] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, pp. 390–404, Apr. 2006.

[12] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, Nov. 2000.

[13] Z. Miao and A. Ortega, "Expected run-time distortion based scheduling for delivery of scalable media," in *Proc. Int'l Packet Video Workshop*, vol. 1, Apr. 2002.

[14] P. Chou and A. Sehgal, "Rate-distortion optimized receiver-driven streaming over best-effort networks," in *Proc. Int'l Packet Video Workshop*, vol. 1, Apr. 2002.

[15] J. Chakareski, J. Apostolopoulos, S. Wee, W. Tan, and B. Girod, "Rate-distortion hint tracks for adaptive video streaming," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, pp. 1257–1269, Oct. 2005.

[16] C. D. Vleeschouwer, J. Chakareski, and P. Frossard, "The virtue of patience in low-complexity scheduling of packetized media with feedback," *IEEE Trans. Multimedia*, vol. 9, pp. 348–365, Feb. 2007.

[17] J. Chakareski, P. Chou, and B. Girod, "Rate-distortion optimized streaming from the edge of the network," in *Proc. Workshop on Multimedia Signal Processing*, Dec. 2002.

[18] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Proc. Data Compression Conference*, Mar. 2003.

[19] W. Li, "Overview of fine granularity scalalability in MPEG-4 video standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, pp. 301–317, Mar. 2001.

[20] H. Wang and A. Ortega, "Robust video communication by combining scalability and multiple description coding techniques," in *Proc. Symp. Electronic Imaging*, vol. 1, Jan. 2003.

[21] G. Cheung and W. Tan, "Directed acyclic graph based source modeling for data unit selection of streaming media over QoS networks," in *Proc. Int'l Conf. Multimedia and Exhibition*, vol. 1, Aug. 2002.

[22] H. Wang and A. Ortega, "Rate-distortion based scheduling of video with multiple decoding paths," in *Proc. Symp. Electronic Imaging*, Jan. 2005.

[23] P. Chou, H. Wang, and V. Padmanabhan, "Layered multiple description coding," in *Proc. Int'l Packet Video Workshop*, vol. 1, Apr. 2003.

[24] L. Kondi, "A rate-distortion optimal hybrid scalable/multiple-description video codec," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, pp. 921–927, July 2005.

[25] M. Kalman and B. Girod, "Rate-distortion optimized streaming of video with multiple independent encodings," in *Proc. Int'l Conf. Image Processing*, Oct. 2004.

[26] B. Birney, May 2003. Intelligent Streaming [Online]. Available: http://www.microsoft.com/windows/windowsmedia/howto/articles/intstreaming.aspx.

[27] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the internet," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, pp. 269–281, Mar. 2001.

[28] S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error resilience support in h.263+," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, pp. 867–877, Nov. 1998.

[29] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," in *Proc. Visual Communications and Image Processing*, pp. 392–409, Jan. 2001.

[30] J. Chakareski, S. Han, and B. Girod, "Layered coding vs. multiple descriptions for video streaming over multiple paths," *ACM/Springer Multimedia Systems Journal*, vol. 10, pp. 275–285, Apr. 2005.