DIRECTED ACYCLIC GRAPH BASED SOURCE MODELING FOR DATA UNIT SELECTION OF STREAMING MEDIA OVER QOS NETWORKS

Gene Cheung, Wai-tian Tan

Hewlett-Packard Laboratories

ABSTRACT

Regardless of the network loss process, a central problem in rate-distortion optimized video streaming is the modeling of the resulting distortion associated with the loss of different subsets of data units. When the dependency among data units is modeled by a directed acyclic graph, previous work has modeled the distortion contribution of a data unit based on only two events: whether or not the data unit and all its dependent data units are available. The restriction to only two events allows the use of a single scalar to represent the distortion contribution of a data unit, and at the same time limits its accuracy. In this paper, we consider the general case when a data unit can assume different distortion contributions when different subsets of its dependent data units are available. Rate-distortion optimized streaming using the general model is performed to demonstrate the possible gains.

1. INTRODUCTION

In this paper, we present a scheme to perform rate-distortion optimization for streaming media over a QoS network. Specifically, we consider the *data unit selection problem*: given a set of encoded media data units, what is the optimal subset of data units that we should send at a given time, and at what network service for each selected data unit, given a rate constraint? Rate constraint is needed so that for unregulated network, the media flow performs proper network congestion control [1], and for regulated network, the flow does not exceed the contracted rate limit. Optimality here is defined as the expected media quality at the client given observable network conditions. We consider both scenarios where the client may or may not be sending feedbacks informing the sender which data units have been correctly received.

The key in formulating the data unit selection problem is the modeling of the encoded source and the network. In this paper, we present a new source model based on directed acyclic graph (DAG), extending an idea originated from [2]. In [2], the distortion reduction associated with every data unit is modeled using two events only: whether or not that data unit and all its dependent data units are available. The implicit assumption is that the marginal usefulness of a data unit is dependent only on whether at least one of the dependent data units is lost, but not on which subset of the dependent data units is lost. This assumption is generally not true and limits the accuracy of the source model. In contrast, we consider a general source model where the marginal distortion reduction of a data unit is a function of its available dependent data units. Despite the increase in modeling complexity, we show how the source model can nevertheless be used in conjunction with a network model to formulate a tractable data unit selection problem.

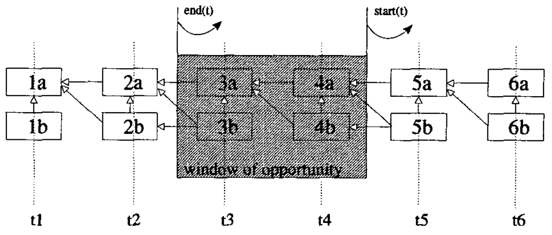


Fig. 1. Data unit Selection in Window of Opportunity

2. RELATED WORK

The data unit selection problem has been studied for the special case for layered coding [3], [4], [5]. While each provides a novel solution, our source modeling is more general and hence has wider applicability beyond layered coding. To a large extent, our work leverages on work in [2], who, to the best of our knowledge, is the first to model dependencies between media data units using DAGs. We differ in that the descriptive power of DAG is more thoroughly exploited in our case to describe the dependencies among source data units. [6] also expands on the source abstraction of [2] to include effects of error concealment by increasing the number of error events and resulting distortions. Unlike [6], we make the dependencies among data units explicitly using DAGs and present an optimization based on such dependencies.

3. SOURCE MODEL: DIRECTED ACYCLIC GRAPH

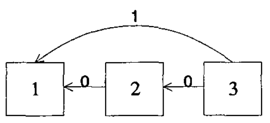
Leveraging abstraction introduced in [2], we model the source coding as a group of data units connected by a directed acyclic graph. A data unit is a logical source coding unit such as a P-frame or a group of macroblocks. Each data unit i (DU_i) is possibly connected to other data units DU_j's via edges k ($i \xrightarrow{k} j$), indicating the distortion reduction of DU_i depends on DU_j's (to be discussed in details later). Each DU_i has a data size in bytes r_i and a delivery deadline T_i, indicating that DU_i arriving at the client after T_i is late and useless. Figure 1 shows a DAG description of a layer-coded media stream, where Xa is the base layer of frame X, and Xb is the enhancement layer of frame X.

3.1. Decoding Edges & Scalar Distortion Nodes

The crux of the matter is how the data units are connected — how the data units are related in terms of decoding correctness and distortion performance. Given the model is a graph, this information is expressed by the graph's edges and nodes. We first discuss what

Decoding Edge	d_1	d_2	case when inaccurate
N	δ_1	δ_2	only DU_1 is delivered
N	$\delta_1 + \gamma_{1,2}$	$\delta_2 - \gamma_{1,2}$	only DU_2 is delivered
Y	δ_1	δ_2	only DU_2 is delivered
Y	$\delta_1 + \gamma_{1,2}$	$\delta_2 - \gamma_{1,2}$	only DU_2 is delivered

Fig. 2. 2 I-frame Example of Scalar Distortion Nodes



error event & distortion for DU3

error event	arrow 1	arrow 0	distortion
0	0	0	$d3^{(0)}$
1	0	1	$d3^{(1)}$
2	1	0	$d3^{(2)}$
3	1	1	$d3^{(3)}$

Fig. 3. General DAG

we term *decoding edges* and *scalar distortion nodes* as described in [2]. We then discuss *delivery edges* and *vector distortion nodes*, which can be viewed as generalizations of the former.

In [2], the dependencies among data units are described by what we term *decoding edges*, where DU_i pointing to a set of DU_j's means DU_i is decoded correctly iff all DU_j's are also *decoded* correctly and DU_i is delivered correctly. In such case, DU_i marginally reduces distortion at the client by d_i. Otherwise, DU_i contributes 0. Whether each DU_j is decoded correctly depends in turn on the set of data-units it is pointing to. Since performance of each data unit is characterized by a single number d_i, we call each node representation of data unit a *scalar distortion node*, or simply a *scalar node*. Edges are drawn based on the characteristics of a particular coding scheme – for example, reference frames point to referee frames. Figure 1 shows the decoding edges for a layer coded media stream.

While decoding edges and scalar nodes lead to a simple and elegant formulation of the data unit selection problem, it has shortcomings when dealing with error concealment. The reason is that decoding edges stemming from DU_i defines only one successful decoding event, leading to one performance number d_i. No partial construction is possible. To see the limitation, consider the simple case when there are only two frames in a media sequence and both are coded as I-frames. We do not draw a decoding edge from DU_2 to DU_1 since decoding of the second I-frame does not depend on the first. Assigning distortion values to d_1 and d_2 is now problematic with error concealment. If we assign d_1 to be the distortion reduction of only frame 1, δ_1, and likewise for d_2, then if only frame 1 is delivered correctly, then we only get $d_1 = \delta_1$ distortion reduction. However, surely we can display frame 1 as frame 2 as a first order approximation with distortion reduction $\gamma_{1,2}$. Yet this simple error concealment cannot be accounted for in the abstraction. Adding a decoding edge between them does not solve the problem either, as shown in Figure 2 where all possibilities using scalar nodes are enumerated.

This simple example shows that scalar nodes with one distortion number for each data unit breaks down even for very simple error concealment strategies. This motivates us to develop a more general abstraction to model media source.

3.2. Delivery Edges & Vector Distortion Nodes

We first introduce a general notion of data unit performance: given DU_i points to a set of DU_j's, a particular combination of correctly decoded DU_j's — a *decoding event* k — means correct delivery of DU_i has distortion reduction $d_i^{(k)}$. Let $\mathcal{V}_i$ be the set of parent data

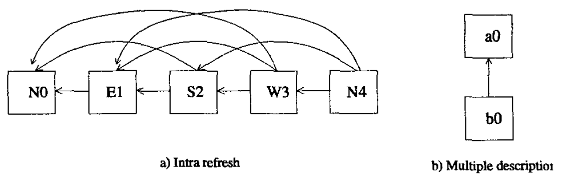


Fig. 4. Examples of Delivery Edges & Vector Nodes

units of DU_i. To completely describe the distortion reduction of DU_i for all decoding events, we require a vector $\mathbf{d}_i$ of length $2^{|\mathcal{V}_i|}$. The k-th component of $\mathbf{d}_i$, $d_i^{(k)}$, is the resulting distortion reduction in decoding event k. When we use one bit to represent the reception of each parent data unit, the decoding event k corresponds to the event where DU_x, $DU_i \xrightarrow{s} DU_x$, is decoded correctly iff sth bit of k is 1. Mathematically, we write $(k \gg s)\&1 = 1$, for $s \in \{0, \dots, |\mathcal{V}_i| - 1\}$. Data unit representation using vector $\mathbf{d}_i$ is called a *vector distortion node*, or simply a *vector node*.

Now that the notion of "correctly decoded" has been generalized from binary events in [2], we introduce a more general notion of dependency: a DU_i points to a set of DU_j's, each of whose *delivery* affects DU_i ability to reduce distortion. In other words, a particular combination of correctly delivered DU_j's — a *delivery event* k — means correct delivery of DU_i has distortion reduction $d_i^{(k)}$. Delivery event k is then the event where DU pointed by edge s is delivered correctly iff the sth bit of k is 1. These new dependency edges are called *delivery edges*. Unlike "correctly decoded", the notion of correct delivery cannot be more general since data units are the atomic units of the source representation.

As an example of delivery edges and vector nodes, consider the DAG in Figure 3, where DU_3 depends on DU_1 and DU_2. In error event 2, DU_1 is delivered correctly but DU_2 is not, and receiving DU_3 correctly can reduce distortion by $d_3^{(2)}$.

Going back to the 2 I-frame example discussed earlier, we can now accurately model the error concealment scheme here by first drawing a delivery edge from DU_2 to DU_1. We then assign $\mathbf{d}_1 = [\delta_1 + \gamma_{1,2}]$, and $\mathbf{d}_2 = [\delta_2,\ \delta_2 - \gamma_{1,2}]$.

Before we discuss the formulation of the data unit selection problem using our developed abstraction, let's consider two more examples where previously proposed decoding edges and scalar nodes are insufficient. Consider first the intra-fresh example in Figure 4a. In this case, each frame is divided into 4 quadrants (N, E, S, W). Frames are assigned to one quadrant label in rotation, and each frame set all macroblocks in the labeled quadrant to I-blocks. This has similar effects as one I-frame for every three P-frames. Using delivery edges, we can express how each frame is depended on the delivery of only the last three frames. This is not possible with decoding edges.

Consider now the simple multiple description example of Figure 4b. Suppose frame 0 is coded into two independently decodeable descriptions $a0$ and $b0$. Each description can be used as an approximation for the other. Despite this mutual relationship, we can accurate model this using vector nodes without causing a cycle in DAG. As shown in Figure 4b, we can connect $b0$ to $a0$, and use $\mathbf{d}_{a0} = [\delta_{a0} + \gamma_{a0,b0}]$ and $\mathbf{d}_{b0} = [\delta_{b0} + \gamma_{b0,a0},\ \delta_{b0} - \gamma_{a0,b0}]$. This is not possible with scalar nodes.

4. DATA UNIT SELECTION PROBLEM

4.1. Problem Framework

Given the source model, the framework under which we formulate the data unit selection problem is the following. We first layout data units in DAG from left to right in delivery deadline order as shown in Figure 1. At each data unit selection instance, we have to make selection decision on data units within a time-varying window called *window of opportunity* (WOOP), defined by start time $start(t)$ and end time $end(t)$. The selection process is repeated every P seconds, with $start(t)$ and $end(t)$ advancing in time resulting in a new WOOP. Each WOOP at time t would be subject to a rate constraint — the admissible sending rate in P seconds, dictated by end-to-end congestion algorithms [1] for unregulated networks, or by pre-arranged service level agreement.

$end(t)$ essentially eliminates data units that are already late or have little chance of arriving at the client on-time given current time t and the observable network delay. $start(t)$ can be set in a number of ways as described in [2]. We will assume functions $end(t)$ and $start(t)$ are both given and will focus on the resulting optimization instead.

4.2. Formulate Optimization

We formulate the data unit selection problem as an optimization problem. Given a DAG of N data units with deadlines falling within WOOP, the problem is to decide which data units to send using what network services. We represent the sequence of decisions as a selection vector of length N, $\pi = [\pi_1, \pi_2 \dots, \pi_N]$, where $\pi_i \in \{0, 1, \dots, K\}$ indicates which one of K network services is DU_i selected for. We will use the convention that $\pi_i = 0$ indicates DU_i is not selected for transmission.

We formulate it as a rate-distortion optimization problem:

$$\min_{\pi} D(\pi) \quad \text{s.t.} \quad R(\pi) \leq R_t \tag{1}$$

where $D(\pi)$ and $R(\pi)$ are the expected distortion and rate constraint respectively given selection vector π.

For network without differentiated service, $\pi_i \in \{0, 1\}$, and the rate constraint is simple:

$$R(\pi) = \sum_{i=1}^{N} \pi_i r_i \tag{2}$$

In the case where there are multiple network services available, instead of having multiple constraints, we envision a simple "cost" constraint where employing better QoS service will incur higher cost per byte. The cost function that maps QoS service type to cost per byte is $c(\pi_i), \pi_i \in \{0, \dots, K\}$, with $c(0) = 0$. The rate constraint in this case is really a cost constraint:

$$R(\pi) = \sum_{i=1}^{N} c(\pi_i) r_i \tag{3}$$

To formulate $D(\pi)$, we need to first discuss the network model we are employing.

4.3. Network Model

We can select $end(t)$ carefully to rule out packets with tight time constraints — each packet in WOOP has sufficient time for delivery to the client. This means each packet is either dropped in the network or arrives on time, and so we only need to model delay accurate enough to set $end(t)$ properly. The consequence is that we can afford a simple network model: we assume K i.i.d. packet loss channels with packet loss probability $\alpha(k)$ for network service $k \in \{1 \dots K\}$. To simplify the discussion, we assume for now a one-to-one mapping between data units and packets.

Let $\phi_i = \{\mathbf{h}_i, a_i\}$ be the history of DU_i in all previous optimization instances — i.e. the number ($|\mathbf{h}_i|$) and transmission class of packet i in previous windows $(h_i^{(1)}, \dots, h_i^{(|\mathbf{h}_i|)})$ and the total number of ACKs received from client (a_i). Further, let $\epsilon(\phi_i, \pi_i)$ be probability that no transmission of packet i is successful given ϕ_i and π_i. Assuming the simple network model above, we can calculate $\epsilon(\phi_i, \pi_i)$ as follows:

$$\epsilon(\phi_i, \pi_i) = \begin{cases} 0 & \text{if } a_i > 0 \\ \alpha(\pi_i) \prod_{h \in \mathbf{h}_i} \alpha(h) & \text{o.w.} \end{cases} \tag{4}$$

4.4. Defining Distortion

Given the network model, we can now define the expected distortion $D(\pi)$ as follows. Let $\mathbf{p}_i(\pi)$ be the probability vector containing error probabilities $\epsilon(\phi_i, \pi_i)$ for DU_j's in $\mathcal{V}_i$ and DU_i. The probability of delivery event k for DU_i, $p_i(\pi)^{(k)}$, is:

$$\begin{aligned} p_i(\pi)^{(k)} &= \prod_{\forall x \in \mathcal{A}} [1 - \epsilon(\phi_x, \pi_x)] \prod_{\forall y \in \mathcal{B}} \epsilon(\phi_y, \pi_y) && (5) \\ \mathcal{A} &= \{x \mid DU_i \overset{s}{\to} DU_x, (k \gg s)\&1 = 1\} && (6) \\ \mathcal{B} &= \{y \mid DU_i \overset{s}{\to} DU_y, (k \gg s)\&1 = 0\} && (7) \end{aligned}$$

where $\mathcal{A}$ and $\mathcal{B}$ are the sets of dependent data units of DU_i that are received and lost respectively. More precisely, DU pointed to by edge s is received correctly (incorrectly) iff sth bit of k is 1 (0).

For a clearer presentation, we define a new operator $\circ$ that sums up all products of probabilities of delivery events and corresponding distortion:

$$\mathbf{p}_i(\pi) \circ \mathbf{d}_i = \sum_{k=0}^{2^{|\mathcal{V}_i|}-1} p_i(\pi)^{(k)} d_i^{(k)} \tag{8}$$

The total distortion for the data unit group $D(\pi)$ can now be defined as:

$$D(\pi) = \sum_{i=1}^{N} \mathbf{p}_i(\pi) \circ \mathbf{d}_i \tag{9}$$

The optimization problem is to minimize distortion in (9) using selection vector π, while subject to constraint (2) or (3). We briefly discuss how this is solved in the next section.

5. OPTIMIZATION SOLUTION

Instead of solving (1) directly, we use the conventional Lagrangian approach [2] and solve the corresponding Lagrangian instead, so that the optimal Lagrangian solution is optimal to (1) up to a convex-hull approximation:

$$\begin{aligned} &\min_{\pi} && D(\pi) + \lambda R(\pi) && (10) \\ &= \min_{\pi} && \textstyle\sum_{i=1}^{N} \mathbf{p}_i(\pi_i) \circ \mathbf{d}_i + \lambda c(\pi_i) r_i && (11) \end{aligned}$$

The problem of finding the appropriate λ for (11) has been studied extensively [7] [2]. We focus instead on solving (11) given λ.

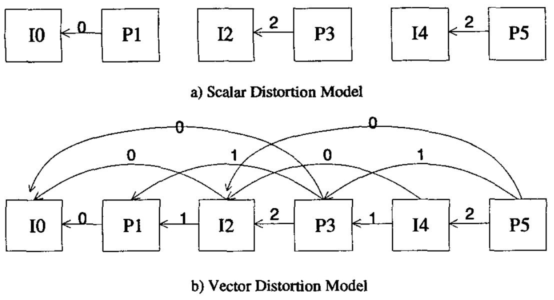


Fig. 5. DAGs used for the Experiment

5.1. Optimization Algorithm

One way to solve (11) given λ is to leverage on the mode-selection optimization algorithm in [8]. The optimization is basically a two-step procedure. Given a group of N data units in a DAG, we first order them totally by renaming them from 1 to N such that if a delivery edge stems from DU_i to DU_j, then $j < i$. Then in the assigned order, we map each DU_i to a stage with appropriate numbers of states, where each state represents a particular permutation of QoS assignments to DUs. Two states from two neighboring stages are connected iff the two enumerations of QoS assignment to DUs that are common to both stages agree. Upon the construction of the state diagram, the optimal solution to (11) is the shortest path from any state in the leftmost stage to any state in the rightmost stage. See [8] for details of the optimization algorithm.

Why is this optimal? The key is to observe that when evaluating the operational distortion for DU_i, we do not need to know QoS assignments of DU beyond $\mathcal{V}_i$ and DU_i itself. If we perform the sum in (11) in an order such that QoS assignments of parent DUs are always pre-enumerated for the corresponding children, then the loosely dependent nature of data units is apparent, and by marching through a trellis that cleverly enumerates the solution space, we can find an optimal solution just like mode-selection for macrobblocks in [8].

The complexity in [8] is high when $|\mathcal{V}_i|$ is large: $\mathcal{O}(N(K+1)^{2|\Theta_{\max}|})$, where $\Theta_{\max} \geq \max_{i=1}^{N} |\mathcal{V}_i|$. For our experiment, we use instead a greedy method where the data unit most immediately profitable in (9) is selected iteratively until the budget (2) is met.

6. EXPERIMENTAL RESULTS

To demonstrate the utility of our developed source modeling — delivery edges and vector nodes, we compare the performance of our model (vector model) against the source model in [2] (scalar model) for identical source and network conditions. For encoded source, we use an H.263 ended video stream of the 50-frame `carphone` sequence with alternating I and P frames. It is coded in QCIF at 20fps, resulting in 332kbps. Using the scalar model, we construct the DAG in Figure 5a.

Alternatively, we use our proposed vector model and constructed the DAG shown in Figure 5b. To construct the DAG, we assume error concealment of each pair of I+P frames depends only on the delivery of the previous pair of I+P frames.

The network simulation is conducted in network simulator 2 (ns2 [9]). The network simulation we constructed is a simple two-node topology, with streaming server at one node and the client at the other. The two nodes are connected by a fixed delay, lossy link with packet loss rate $\alpha(1)$ — only one network service is available. The two DAGs with associated distortion numbers in the two source models are loaded into ns2 for data unit selection at the server side. The optimization procedure described in Section 5 is performed there. Optimization window WOOP is a fixed 10-frame window, and optimization period P is 5-frame time or 0.25 seconds. We assume each data unit (frame) is packed into one packet, with bandwidth at each optimization instance fixed at 6 packets.

At the client, actual decoding is simulated as follows. An I-frame is deemed "correctly decoded" if the corresponding I-frame data unit is delivered correctly, and a P-frame is deemed "correctly decoded" if the corresponding P-frame data unit and the previous I-frame data unit are both delivered correctly. If a frame i is correctly decoded, then the PSNR is δ_i. If a frame i is not correctly decoded, then the most recently correctly decoded frame j is used for display, resulting in PSNR $\gamma_{j,i}$. If no such frame is available, then the PSNR is simply 0.

Results comparing the two source models under two scenarios — with and without client's ACK packets, are shown in Fig. 6. The gain in PSNR using our more accurate model ranges from $0.16dB$ to $0.53dB$. Notice the pattern that as the network condition worsens, the difference in performance between the two models increases. This is expected, since the improved model better captures the effects of error concealment, which is more important when network conditions worsen.

Scheme	$\alpha(1) = 0.05$	$\alpha(1) = 0.1$	$\alpha(1) = 0.15$
Vector w/ ACK	37.10	36.65	36.11
Scalar w/ ACK	36.94	36.31	35.63
Vector w/o ACK	37.07	36.54	36.10
Scalar w/o ACK	36.81	36.01	35.57

Fig. 6. Comparison of Results in PSNR

7. REFERENCES

[1] S. Floyd et. al., "Equation-based congestion control for unicast applications," in *SIGCOMM*, 2000.

[2] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," in *submitted to IEEE Trans. MM*, February 2001.

[3] M. Podolsky, S. McCanne, and M. Vetterli, "Soft arq for layered streaming media," Tech. Rep. UCB/CSD-98-1024, University of California, Berkeley, November 1998.

[4] V. Chande and N. Farvardin, "Progressive transmission of images over memoryless noisy channels," in *IEEE JSAC*, June 2000, vol. 18, no.6.

[5] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Asilomar*, 2000.

[6] R. Zhang et. al., "End-to-end distortion estimation for rd-based robust delivery of pre-compressed video," in *Asilomar*, 2001.

[7] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," in *IEEE SP Magazine*, November 1998.

[8] G. Cheung, "Directed acyclic graph based mode optimization for h.263 video encoding," in *ICIP*, 2001.

[9] "The network simulator ns-2," June 2001, release 2.1b8a, http://www.isi.edu/nsnam/ns/.