Real-Time Imaging 7, 255–273 (2001)
doi:10.1006/rtim.2001.0226, available online at http://www.idealibrary.com on IDEAL®

Efficient Selective Frame Discard Algorithms for Stored Video Delivery Across Resource Constrained Networks

Video delivery from a server to a client across a network is an important component of many multimedia applications. While delivering a video stream across a resource constrained network, loss of frames may be unavoidable. Under such circumstances, it is desirable to find a server transmission schedule that can efficiently utilize the network resources while maximizing the perceived quality-of-service (QoS) at the client. To address this issue, in this paper we introduce the notion of *selective frame discard* at the server and formulate the *optimal selective frame discard* problem using a QoS-based cost function. Given network bandwidth and client buffer constraints, we develop an $O(N \log N)$ algorithm to find the minimum number of frames that must be discarded in order to meet these constraints. The correctness of the algorithm is also formally established. We present a dynamic programming based algorithm for solving the problem of optimal selective frame discard. Since the computational complexity of the optimal algorithm is prohibitively high in general, we also develop several efficient heuristic algorithms for selective frame discard. These algorithms are evaluated using JPEG and MPEG video traces.

Zhi-Li Zhang*,[1], Srihari Nelakuditi[1], Rahul Aggarwal[1] and Rose P. Tsang[2]

[1]*Dept. of Computer Science & Engineering, University of Minnesota, Minneapolis, MN55455, E-mail: {zhzhang, srihari, raggarwa}@cs.umn.edu*
[2]*Sandia National Laboratories, PO Box 969, Mail Stop 9011, Livermore, California 94550, E-mail: rtsang@ca.sandia.gov*

Introduction

The playback of stored video over a network is required by several applications such as digital libraries, distance learning and collaboration, video and image servers and interactive virtual environments. Stored video typically has high bandwidth requirements and exhibits significant rate variability [1–3]. This is particularly the case when variable bit rate encoding schemes are used. In a network where resources such as the network bandwidth and buffering capacity are constrained, it is a major challenge to design an efficient stored video delivery system that can achieve high resource utilization while maximizing users' perceived quality-of-service (QoS).

Video smoothing techniques [4–11] have been proposed for reducing the network bandwidth requirement of bursty video streams by taking advantage of client buffering capabilities. Similar techniques have also been developed when network bandwidth is constrained instead of the client buffer [12–15]. In reality, however, both network bandwidth and client buffering capacity

*Author to which all correspondence should be addressed.

are likely to be limited. Under such circumstances, there may *not* be a feasible server transmission schedule that can deliver video streams to clients without incurring loss of data. Instead of being denied service, clients may choose to receive lower quality video streams with occasional frame losses. This may arise, for example, in the case of constant-bit-rate (CBR) service, where for a client with a limited buffer, the network may not have sufficient bandwidth to support the peak rate of a smoothed video stream, or in the case of renegotiated CBR (RCBR) service [16], where bandwidth renegotiation fails in the middle of a video transmission.

When delivering a video stream across a resource-constrained network, a naive approach at the server may attempt to transmit each frame with no awareness of the resource constraints. As a result the network may drop packets causing frame losses. In addition, the client may be forced to drop frames that arrive too late for playback. This results in wastage of network bandwidth and client buffer resources. In this paper, we introduce the concept of *selective frame discard*[2] *(SFD)* at the server which *pre-emptively* discards frames in an intelligent manner by taking network constraints and client QoS requirements into consideration. The proposed server selective frame discard has two advantages. First, by taking the network bandwidth and client buffer constraints into account, the server can make the best use of network resources by selectively discarding frames in order to minimize the likelihood of future frames being discarded, thereby increasing the overall quality of the video delivered. Second, unlike frame dropping at the network or the client, the server can also take advantage of *application-specific* information such as information content of a frame and inter-dependencies, in its decision in discarding frames. As a result, the server optimizes the perceived quality of service at the client while maintaining efficient utilization of the network resources.

In this paper, we develop various selective frame discard algorithms for stored video delivery across a network where both the network bandwidth and the client buffer capacity are limited. We begin by formulating the problem of *optimal selective frame discard* using the notion of a cost function. The cost incorporates the QoS metrics of clients. Given network bandwidth and client buffer constraints, we develop an $O(N \log N)$ algorithm to find the minimum number of frames that must be discarded in order to meet these constraints. The correctness of the algorithm is also formally established. We then present a dynamic programming based algorithm for solving the problem of optimal selective frame discard. This algorithm computes a transmission schedule that optimizes the client QoS based on a given cost function. Since the computational complexity of the optimal algorithm is prohibitively high in general, we also develop several efficient heuristic algorithms which take both resource constraints and cost into consideration. These algorithms are evaluated using JPEG and MPEG video traces. Through the performance evaluation, we find that the proposed *minimum cost maximum gain* heuristic algorithm yields near-optimal performance for both JPEG and MPEG encoded video.

The rest of this paper is organized as follows. In the next section, we briefly discuss related work. The section following describes the problem setting and formulates the optimal selective frame discard problem. The minimum frame discard algorithm is described and its correctness is then proved. An optimal dynamic programming based selective frame discard algorithm is then presented. Then, several efficient selective frame discard heuristics are introduced and a performance evaluation based on JPEG traces is presented. These algorithms are extended to MPEG in the penultimate section. The last section concludes the paper.

Related work

Rate adaptation and control for compressed video has been extensively studied, in particular, in the context of joint source and channel adaptive encoding (see, e.g., [17–23]). For example, in [19,23], the problem of finding an optimal transmission schedule with leaky bucket constraints is studied with some form of cost functions. In [18, 20, 21], video rate adaptation through quantization parameter adjustment over networks with feedback-based congestion control mechanisms is discussed. These techniques are mostly designed for *live* video transmission, and may not be suitable for delivery of stored video, for a number of reasons (e.g., absence of encoders at a stored video server, or the processing overhead/delay incurred).

[2]In this paper we assume that frames are basic *application-level* data units for server selective discard. This assumption is not necessary. The algorithms developed in the paper do not hinge on this assumption. In practice, other (preferably) application-level data units such as slices, blocks or macro blocks in JPEG and MPEG can also be used as the basis for server selective discard.

Packet discarding schemes which take advantage of application specific information have been used in many different contexts. For example, in [24] a frame-induced packet discarding scheme at the network level is introduced. In this scheme, upon detection of loss of a threshold number of packets belonging to a video frame, the network attempts to discard all the remaining packets of that frame. Similarly, the continuous media toolkit (CMT) [25, 26] also uses application-specific information to assign priority in packet discarding. Our problem setting, however, is considerably different from these existing studies. In designing efficient server selective frame discard algorithms, we leverage application-specific information to optimize the client QoS while at the same time taking both network bandwidth and client buffer constraints into account.

Problem Formulation

In this section we provide an overview of the stored video delivery system and motivate the notion of *selective frame discard* at the server for a resource constrained network. The idea of a cost function is introduced to incorporate a QoS metric and is used to formulate the selective frame discard problem.

Figure 1 depicts a server transmitting a stored video stream to a client across a network. The video data is retrieved from the disk subsystem into the server memory and moved onto the network as per some server transmission schedule. The client has a buffer which can be used for the work ahead of video data by the server. The client plays back the video frames periodically as determined by the frame rate. Each video frame has a deadline constraint associated with it. Since the frames are being played back at a periodic rate, the frame has to be available at the client when the decoding process attempts to display it. If the frame is not available, the playback is paused, resulting in a *playback discontinuity*.

We assume a network environment where a fixed amount of network bandwidth can be reserved for a video stream (i.e., the CBR service). However, the network resource may be constrained (this constraint is referred to as *rate constraint*). Furthermore, the buffer resource at a client may also be constrained (this constraint is referred to as *client buffer constraint*). In such a resource constrained system, there may not be sufficient resources to ensure the continuous playback of the video at the client. While the rate constraint

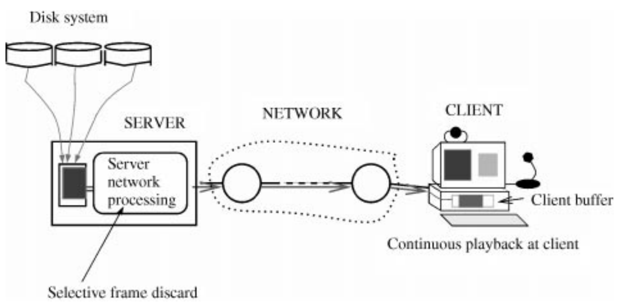


Figure 1. Overview of the problem setting.

regulates the amount of data that can be transmitted in one time unit, the client buffer constraint limits the amount of work ahead by the server into the client buffer. In the presence of both rate and buffer constraints, a feasible server transmission schedule which satisfies both constraints simultaneously may not exist. Hence in these circumstances, frame dropping is unavoidable.

A naive approach at the server may attempt to transmit each frame with no perception of the resource constraints. This may cause packet loss and delay in the network or buffer overflow at the client. As a result, the client may receive incomplete frames which cannot be played back. Also, the client may be forced to drop a frame if it arrives late. The system resources consumed by these dropped frames are effectively wasted.

Selective frame discard aims at optimizing the utilization of the network resources by *pre-emptively* discarding frames at the server. A frame is transmitted only if it can meet its playback deadline; Otherwise, the frame is discarded thereby increasing the likelihood of other frames meeting their playback deadlines. By effectively utilizing the resources, selective frame discard improves the playback continuity.

In formulating the selective frame discard problem, we consider a discrete-time model at the frame level. Each time slot represents the unit of time for playing back a video frame. For simplicity of exposition, we assume zero startup delay, i.e., the time the server starts video transmission and the time the client starts playback is the same. We also ignore the network delay. Table 1 summarizes the notation we introduce in this section.

Table 1. Notation

N	:	Length of video in frames.
f_i	:	size of ith frame.
B	:	client buffer capacity for storing unplayed frames.
C	:	network bandwidth.
$\mathcal{N}$	:	set of all frames, i.e., $\{1, \quad, N\}$
S	:	a subset of frames, i.e., $S \subseteq \mathcal{N}$
$A(S)$	:	a transmission schedule w.r.t. set S
$A_i(S)$	:	cumulative data sent by the server over $[1, i]$
$a_i(S)$	:	amount of data sent by the server in slot i
$D(S)$	:	underflow curve w.r.t set S
$D_i(S)$	:	cumulative data consumed by the client over $[1, i]$
$U(S)$	:	overflow curve w.r.t set S
$U_i(S)$	:	maximum cumulative data that can be received by the client over $[1, i]$
$B_i(S)$	:	buffer occupancy at the end of time slot i.
$\hat{A}(S)$	:	greedy transmission schedule w.r.t set S
$\hat{A}_i(S)$	:	cumulative data sent by the server over $[1, i]$ according to the greedy schedule
$\hat{a}_i(S)$	:	amount of data transmitted in slot i under $\hat{A}(S)$

Consider a video stream with N frames. For $i \in \mathcal{N} = \{1, \quad, N\}$, the size of the i^{th} frame is denoted by f_i. Let C denote the bandwidth of the network (i.e., server transmission rate is limited by C per unit of time), and B is the client buffer size. For $S \subseteq \mathcal{N}$, $1_{\{j \in S\}}$ is the indicator function: $1_{\{j \in S\}} = 1$ if $j \in S$ and 0 if $j \notin S$. Let $D(S) = \{D_0(S), D_1(S), \quad, D_N(S)\}$ where $D_i(S) = \sum_{j=0}^{i} f_j \, 1_{\{j \in S\}}$, and let $U(S) = \{U_0(S), U_1(S), \quad, U_N(S)\}$ where $U_i(S) = D_i(S) + B$. We refer to $D(S)$ as the (*client*) *buffer underflow curve with respect to* S, and $U(S)$ as the (*client*) *buffer overflow curve with respect to* S. A server transmission schedule $A(S)$ associated with S is a schedule which only transmits frames included in S, namely, frame i is transmitted under $A(S)$ if and only if $i \in S$. Let $a_i(S)$ be the amount of video data transmitted during time slot i, $i = 1, \quad, N$. In accordance with the notation for $D(S)$ and $U(S)$, the schedule $A(S)$ is denoted by $A(S) = \{A_0(S), A_1(S), \quad, A_N(S)\}$ where $A_0(S) = 0$ and $A_i(S) = \sum_{j=0}^{i} a_j(S)$. Examples of $D(S)$, $U(S)$ and $A(S)$ are shown in Figure 2.

A server transmission schedule $A(S)$ is said to be *feasible* with respect to S if and only if for $i = 0, 1, \quad, N$, (1) *rate constraint is not violated*, i.e., $a_i(S) \leq C$; (2) *buffer constraint is not violated*, i.e., $A_i(S) \leq U_i(S)$; and (3) *playback constraints are not violated*, i.e, $D_i(S) \leq A_i(S)$. In other words, a transmission schedule is feasible if it lies within the buffer underflow curve $D(S)$ and the buffer overflow curve $U(S)$, having slope no more than C (see Figure 2 for an illustration). A set $S \subseteq \mathcal{N}$ is said to be *feasible* if and only if there exists a feasible transmission schedule $A(S)$ with respect to S. For a given pair of rate and buffer constraints (C, B), we denote the collection of all feasible sets by $\mathcal{SFD}(C, B)$.

Given a schedule $A(S)$, the buffer occupancy at the end of time slot i (namely, immediately after frame i has been retrieved from the client buffer if $i \in S$) is denoted by $B_i(S)$. $B_i(S)$ satisfies the following recurrence relation:

$$B_i(S) = \max\{\min\{B_{i-1}(S) + a_i(S), B\} - f_i \mathbf{1}_{\{i \in S\}}, 0\},$$

where $B_0(S) = 0$.

If $B_{i-1}(S) + a_i(S) > B$, the *buffer overflow* occurs at time i. If $B_{i-1}(S) + a_i(S) < f_i$, then *buffer underflow* occurs at time i. Clearly for a feasible schedule, $B_i(S) = B_{i-1}(S) + a_i(S) - f_i \mathbf{1}_{\{i \in S\}}$.

Associated with each S, we define a special schedule $\hat{A}(S)$, referred to as the *greedy transmission schedule* with respect to S. Under $\hat{A}(S)$, the amount of data transmitted in time slot $i, i = 1, \quad, N$, is given by $a_i(S) = min\{B - B_{i-1}(S), C\}$, where $B_0(S) = 0$ and $B_i(S) = B_{i-1}(S) + a_i(S) - f_i \mathbf{1}_{\{i \in S\}}$. Hence, $\hat{A}(S) = \{A_0(S), A_1(S), \quad, A_N(S)\}$, where $\hat{A}_i(S) = \sum_{j=0}^{i} a_j(S)$. It is clear that $\hat{A}(S)$ transmits at the rate C whenever possible without overflowing the buffer (see Figure 2 for an example). In other words, it attempts to keep the buffer as full as possible. By definition, $\hat{A}(S)$ always lies below the buffer overflow curve $U(S)$. Hence $\hat{A}(S)$ is feasible if it stays above the underflow curve $D(S)$, i.e., if $\hat{A}_i \geq D_i(S)$, $i = 0, 1, \quad, N$.

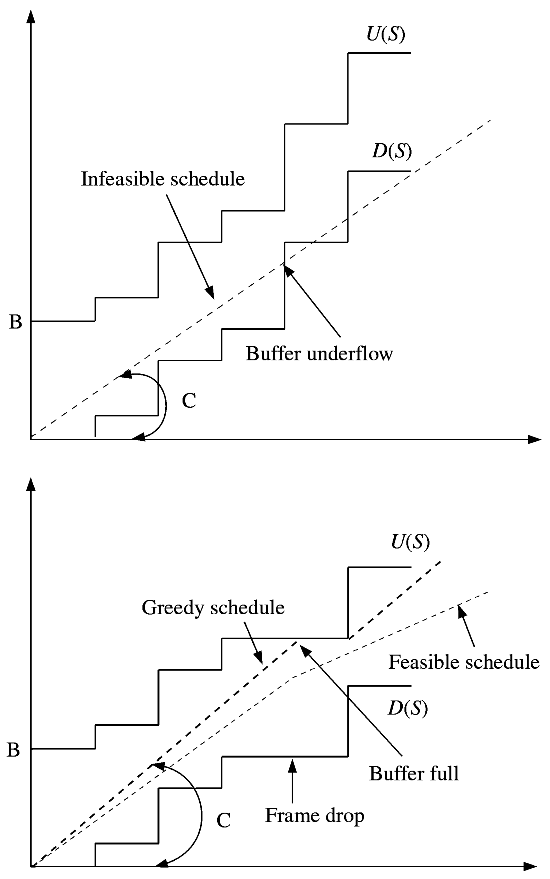


Figure 2. Relation of $D(S)$, $U(S)$ and a server transmission schedule $A(S)$: (a) An infeasible schedule; (b) a feasible schedule.

The greedy schedule $\hat{A}(S)$ has the following property, the proof of which is straightforward.

Proposition 1: *For any* $S \subseteq \mathcal{N}$, *if* $A(S)$ *is a schedule conforming to the rate constraint, then* $A_i(S) \leq A_i(S)$, $i = 0, 1, \quad, N$.

Since $\hat{A}(S)$ bounds the amount of data that can be transmitted from the above, any feasible transmission schedule has to stay below $\hat{A}(S)$. Hence if $\hat{A}(S)$ is not feasible, then any other schedule $A(S)$ is not feasible. As a result, for any $S \subseteq \mathcal{N}$, *S is a feasible set if and only if $\hat{A}(S)$ is feasible with respect to S.*

For a given pair of rate and buffer constraints (C, B), there are in general more than one feasible set. For example, trivially $S = \emptyset$ is always a feasible set. Obviously the perceived quality of the playback at the client would depend on the frames transmitted by the server. It is likely that the greater the number of frames dropped, the lesser the perceived video quality. In addition, consecutive losses of frames or a cluster of lost frames in near proximity would have a more pronounced impact on the perceived video quality than dispersed losses of frames. In order to reflect the perceived video quality at the client, we introduce the notion of a *cost function*, $\phi(S)$, to quantify the "desirability" of different feasible sets. Such a function associates a certain cost with each discarded frame. The cost of a feasible set $\phi(S)$ is the cost associated with the frames that are not part of the set. For an appropriately defined cost function, $\phi(S)$ should reflect the perceived quality of playing back the set S. Thus, minimizing the cost is equivalent to optimizing the QoS at the client.

For a given cost function ϕ, the *optimal selective frame discard problem* therefore is to find a feasible set S^* which minimizes the associated cost $\phi(S^*)$, formally:

Find a set S^* such that $S^* \in \mathcal{SFD}(C, B)$ and

$$\phi(S^*) = \min\{\phi(S) \quad S \in \mathcal{SFD}(C, B)\}$$

S^* is referred to as an *optimal feasible set* with respect to ϕ.

Upper Bound on the Size of Feasible Sets

Before we address the problem of finding an optimal set that minimizes a given cost function, we first consider a more fundamental question:

What is the minimum number of frames to be discarded so that the remaining frames that are transmitted by the server can meet their respective playback time under the known network bandwidth (rate) and client buffer constraints? Or in other words, what is the largest possible cardinality of any feasible set $S \in \mathcal{SFD}(C, B)$?

The solution to this question is not only of interest in its own right, but, as we will see, also sheds light on the design of efficient selective frame discard algorithms in a later section. In this section, we first present an algorithm for solving this problem, and then establish its correctness. This algorithm is referred to as the *minimum frame discard* (in short MINFD) algorithm. In the next section, we present the details of the MINFD algorithm for video streams encoded using an intra-frame encoding scheme such as JPEG, where there is no inter-frame dependence among the frames. Extension of the MINFD algorithm to handle inter-frame dependence is then discussed in the section immediately following, using the MPEG encoding scheme as an example.

The MINFD Algorithm for Intra-Frame Encoded Video

Consider a video stream encoded using an intra-frame encoding scheme such as JPEG. Following the notation introduced in the previous section, f_i denotes the size of the i^{th} frame of the video stream. Let C denote the available network bandwidth (i.e. the rate constraint) and B, the size of the client buffer.

The following observations play a key role in the development of the MINFD algorithm.

1. As long as the buffer constraint is not violated, always try to send as much data as possible (i.e., send at rate C)
2. Whenever the buffer is full, delay transmission until the buffer is no longer completely filled and then resume transmission at rate C. Note that *it is never necessary to discard frames because of buffer overflow*.
3. Whenever a playback deadline cannot be met, either the current frame or an earlier frame must be discarded. This is because the total size of the currently included frames is more than that can be transmitted using the available bandwidth *subject to the buffer constraint*. In deciding the frames to be discarded, we should choose those that would optimize the likelihood of the deadlines of future frames being met.

The first two observations state that we should follow the greedy schedule in transmitting the video data. Based on the third observation, we devise a strategy which discards the frame that maximizes the *buffer occupancy* at the time when a playback deadline is violated. In Theorem 3, we show that this strategy is optimal in the sense that it minimizes the total number of frames discarded.

The algorithm is presented in pseudo-code in Figure 3. It proceeds in stages, $i = 0, 1, \quad, N$, and constructs a feasible set S iteratively.

```
1.  PROCEDURE MINFD (C, B)
2.    Initialization (i = 0): $S^{\#} = \emptyset$, $B_i = 0$, $i_0 = 0$.
3.    For i = 1 to N
4.      $\hat{a}_i = C$
5.      If $B_{i-1} + C > B$, i.e., is buffer full?
6.        $\hat{a}_i = B - B_{i-1}$ and $i_0 = i$
7.      Else
8.        If $B_{i-1} + C < f_i$, i.e., is deadline of frame $i$ violated?
9.          For j = $i_0$+1 to i
10.           Compute the gain $\Delta^i_j$
11.         Choose the frame $k$ with the largest gain max $\Delta_i$
12.         Discard frame $k$ and include frame $i$, i.e.,
13.         $S^{\#} := (S^{\#} \cup \{i\}) \backslash \{k\}$
14.         Update buffer occupancy at $B_i$, i.e.,
15.         $B_i := B_{i-1} + C + \max \Delta_i - f_i$
16.         Update $i_0$ if necessary
17.       Else
18.         $S^{\#} := S^{\#} \cup \{i\}$
19.   Output $S^{\#}$
20. END PROCEDURE
```

Figure 3. The minimum frame discard (MINFD) algorithm.

At stage 0 (line 2 in the algorithm), we start with $S \quad = \emptyset$. At this point, the buffer occupancy $B_0 = 0$. The variable i_0 is used to keep record of the most recent buffer full point if any, and is initialized to 0. At any stage i (lines 3–16), $i = 1, \quad, N$, we follow the greedy schedule $\hat{A}$, and transmit as much data as possible, namely, $a_i = \min\{C, B - B_{i-1}\}$ (lines 4–6). If the buffer is full at this point, set $i_0 = i$. Otherwise (lines 7–16), we check to see whether the playback deadline of frame i is met by the greedy schedule $\hat{A}$ with respect to the current feasible set S (line 8). If $B_{i-1} + C < f_i$, the playback deadline of frame i is violated, a frame needs to be discarded. In order to decide which frame to discard, for each j, $1 \leq j \leq i$, we introduce the notion of *gain* in the buffer occupancy at time i if frame j is discarded. We denote this gain by ^i_j; its definition will be given shortly. The frame discarded, say, frame k, is thus the one which yields the largest gain, namely, $^i_k = \max_{1 \leq j \leq i} \ ^i_j$. This is done in lines 9–11.

Formally, let S_{i-1} denote the feasible set constructed at stage $i - 1$. Recall that $D_j(S_{i-1})$, $U_j(S_{i-1}) = D_j(S_{i-1}) + B$ and $A_j(S_{i-1})$ represent the buffer underflow curve, buffer overflow curve and the amount of data transmitted by the greedy schedule up to time j with respect to S_{i-1}. For $j = 1, \quad, i - 1$, define

$$\nabla^i_j = \min_{j \leq l \leq i-1} \{U_l(S_{i-1}) - A_l(S_{i-1})\}$$

$$= \min_{j \leq l \leq i-1} \{D_l(S_{i-1}) + B - A_l(S_{i-1})\}. \quad (1)$$

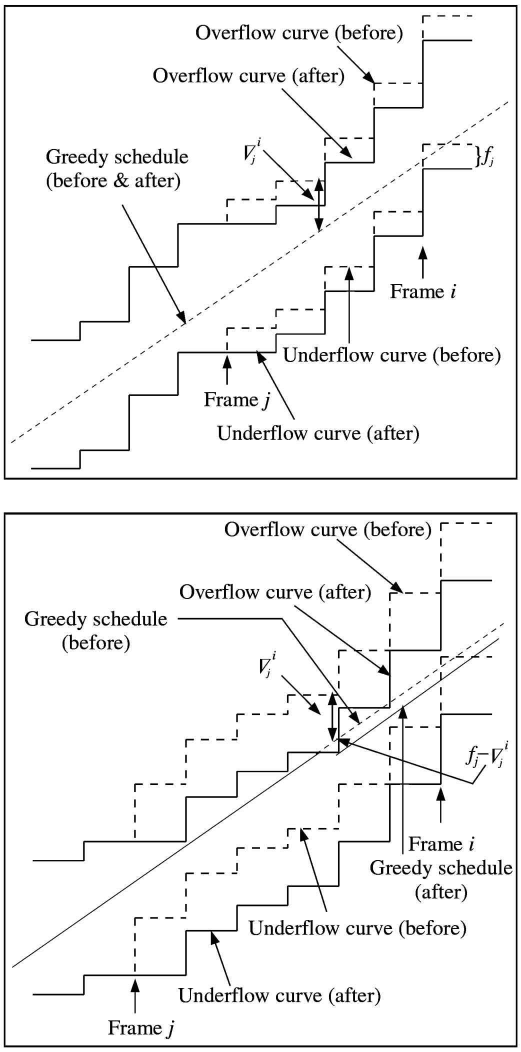


Figure 4. Effect of discarding a frame j on D, U and $\hat{A}$: (a) $f_j < \nabla_j^i$; (b) $fi \geq \nabla_j^i$.

∇_j^i represents the minimal difference between the buffer overflow curve $U(S_{i-1})$ and the greedy schedule $\hat{A}(S_{i-1})$ in the time interval $[j, i-1]$. Intuitively, it is the maximal amount that we can shift the segment $[j, i]$ of $U(S_{i-1})$ downwards towards $\hat{A}(S_{i-1})$ without crossing $\hat{A}(S_{i-1})$ (see Figure 4).

Now we are in a position to define $\ _j^i$.

$$\ _j^i = \begin{cases} f_i, & j = i \\ \min\{f_j, \nabla_j^i\}, & j = 1, \quad , i-1. \end{cases} \tag{2}$$

We now show that $\ _j^i$ is the gain in the buffer occupancy at time i if frame j is discarded. More precisely,

$$B_{i-1}(S_{i-1} \backslash \{j\}) = B_{i-1}(S_{i-1}) + \ _j^i. \tag{3}$$

This is shown pictorially in Figure where the two cases: (a) $f_j \leq \nabla_j^i$ and (b) $f_j > \nabla_j^i$ are depicted. As a result of discarding frame j, the segment $[j+1, i-1]$ of the new buffer overflow curve $U(S_{i-1} \backslash \{j\})$ and underflow curve $D(S_{i-1} \backslash \{j\})$ are the original ones $[U(S_{i-1})$ and $D(S_{i-1})]$ shifted f_j amount downwards. Consider the case where $f_j \leq \nabla_j^i$. This can only occur if $j > i_0$ where i_0 is the last time before i the buffer is full. The greedy schedule can transmit exactly the same amount of data as the original schedule, i.e., $(i-1-j)C$, during the time interval $[j+1, i-1]$. Therefore, (3) holds at time $i-1$. On the other hand, if $f_j > \nabla_j^i$, the amount of data transmitted by the greedy schedule during the same interval is only $(i-1-j)C - (f_j - \nabla_j^i)$. This is because the buffer becomes full at some point. Hence the greedy schedule needs to stop transmission for a duration of $(f_j - \nabla_j^i)/C$ time. Thus, (3) also holds at time $i-1$. Note that in either case, we have:

$$A_{i-1}(S_{i-1} \backslash \{j\}) - A_j(S_{i-1} \backslash \{j\})$$

$$= A_{i-1}(S_{i-1}) - A_j(S_{i-1}) - (f_j - \ _j^i). \tag{4}$$

From (1), $\nabla_j^i = 0$ for any $j \leq i_0$. Therefore, discarding any frame before time i_0 will result in zero gain, i.e., $\ _j^i = 0$. In other words, discarding any frame before the last buffer full point will not help meet the playback deadline of frame i. This is the reason that in line 9 of the algorithm in Figure 3, we only search in the range of $[i_0 + 1, i]$ for a frame to discard. Let k, $i_0 + 1 \leq k \leq i$ be such that $\ _k^i = \max_{i_0+1 \leq j \leq i} \ _j^i$. Hence, discarding frame k yields the maximal gain at time i. Denote this maximal gain by max $\ _i$, i.e., max $\ _i = \ _k^i$. As $\ _i^i = f_i$, we have max $\ _i \geq f_i$. Hence from (3)

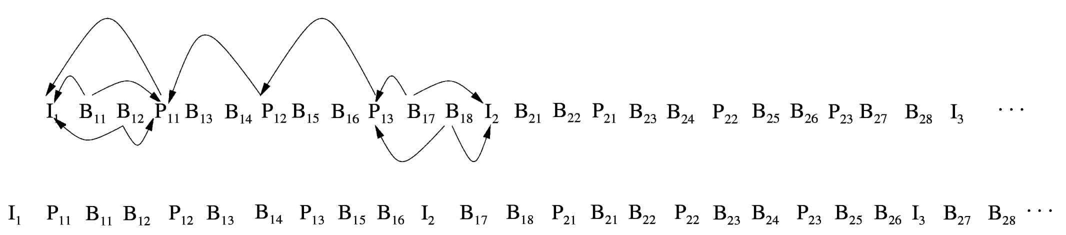
$$B_{i-1}(S_{i-1} \backslash \{k\}) \geq B_{i-1}(S_{i-1}) + f_i.$$

Therefore, if $k \neq i$, discarding frame k will help meet the playback deadline of frame i. As a result of discarding frame k from S_{i-1} and including frame i at stage i, i.e., setting $S_i \ = S_{i-1} \cup \{i\} \backslash \{k\}$ (lines 12–13), we have:

$$B_i(S_i\ ) = B_{i-1}(S_{i-1}) + C + \max \ _i - f_i. \tag{5}$$

Note that the above equation also holds when $k = i$.

In lines 14–16 of the algorithm, the buffer occupancy B_i is updated using (5), and i_0 is set to k^* if discarding k results in a full buffer at time k^*, where k^* is the point where the minimum in (1) is attained. If the deadline of frame i is met, it is included in S_i by setting $S_i \ = S_{i-1} \cup \{i\}$ (line 18). The algorithm stops after stage N and outputs the set S .



I1 B11 B12 P11 B13 B14 P12 B15 B16 P13 B17 B18 I2 B21 B22 P21 B23 B24 P22 B25 B26 P23 B27 B28 I3 ⋯
I1 P11 B11 B12 P12 B13 B14 P13 B15 B16 I2 B17 B18 P21 B21 B22 P22 B23 B24 P23 B25 B26 I3 B27 B28 ⋯

Figure 5. Sequence of MPEG frames with a GOP of size 12. Frame P_{ij}/B_{ij} refers to *j*th P/B frame in the *i*th GOP. (a) Playback order with dependecies (e.g. B_{17} depends on P_{13} and I_2); (b) decoding order.

The feasible set S_i constructed at stage i of the MINFD algorithm has the following important property, the proof of which can be found in the technical report version of this paper [27].

LEMMA 2: *Let S be any feasible set*, i.e., $S \in \mathscr{SFD}(C,B)$. *Then*

$$|S_i| \geq |S \cap \{1,2, \quad ,i\}| \tag{6}$$

where $|\cdot|$ *denote the cardinality of a set.*

Moreover, for any $j = 1,2, \quad ,i$, *if*

$$|S_i \cap \{1, \quad ,j\}| = |S \cap \{1, \quad ,j\}|, \tag{7}$$

then $B_j(S_i) \geq B_j(S)$.

Intuitively, Lemma 2 states that the number of frames included in the (partial) feasible set S_i constructed at stage i is at least as large as the number of frames (up to time i) that are included in any other feasible set. Moreover, among all feasible sets that discard the same number of frames up to time j, S_i maximizes the buffer occupancy at time j. Hence, S_i maximizes the chance of future frames to meet their playback deadlines. As a consequence of this lemma, the transmission schedule S produced by the MINFD algorithm results in the minimum number of discarded frames for any cost function, or equivalently, $|S|$ is maximized.

THEOREM 3: Let S be the feasible set produced by the MINFD algorithm. Then

$$|S^{\#}| = \max\{|S| : S \in \mathscr{SFD}(C,B)\}. \tag{8}$$

By using a clever data structure for maintaining and updating the gain $^{i}_{j}$, we can design an $O(N \log N)$ algorithm to construct S. Due to space limitations, we will not describe it here.

The MINFD Algorithm for MPEG Encoded Video

The MINFD algorithm described in Figure 3 can be extended to handle video streams with inter-frame dependencies such as those encoded using the MPEG encoding scheme. In this section, we illustrate how to modify the MINFD algorithm to handle inter-frame dependencies using MPEG as an example.

The MPEG standard defines three types of frames: I frames, P frames and B frames. I frames are coded autonomously, while P frames are coded with respect to the previous I or P frame. B frames use both previous and future I or P frames as a reference. The MPEG standard also defines a group-of-pictures (GOP) as a consecutive sequence of frames (pictures) containing a single I frame, which is the first frame of the group, and upon which the rest of the frames in the group depend. An example of GOP of size 12 is IBBPBBPBBPBB (see Figure 5(a) for an illustration). Because of the inter-frame dependencies, discarding an I frame results in the loss of an entire GOP. Similarly, discarding a P frame results in the loss of the P and B frames that depend on it.

Another complication arising from these inter-frame dependencies is that the client playback order differs from the client decoding order. Figure 5 illustrates this key difference between the playback order and decoding order for a sequence of MPEG frames with a GOP of size 12. Both the playback order and the decoding order play a role in determining whether a frame can be played back in time or not. For example, whether a B frame can be played back depends not only on its own playback deadline (which is determined by the playback order), but also on the in-time arrival of its future reference frame (which is determined by the decoding order).

We now describe how to extend the MINFD algorithm to handle the inter-frame dependencies in MPEG, using a GOP of size 12 as an example. For simplicity of exposition, we assume that a sequence of MPEG frames are transmitted from the server to the client in their decoding order. We proceed by considering transmission of all or part of the frames in a GOP at each stage. Note that because the last two B frames in a GOP of size 12 depend not only on the previous P frame, but also on the I frame of the next GOP, these two B frames need to be treated specially: they are only included for consideration of transmission in the next GOP (after the I frame of the next GOP, on which they depend), but they are eligible for consideration of discarding in the current GOP. This special treatment will be further expanded on, as we sketch the extended MINFD algorithm for MPEG video below.

Suppose we have a MPEG video sequence of frames with N GOPs[3] of size 12. At stage i, $i = 1, 2, \quad , N$, we consider transmission of the ith GOP, $I_i[B_{i-1,7}B_{i-1,8}]P_{i,1}B_{i,1}B_{i,2}P_{i,2}B_{i,3}B_{i,4}P_{i,3}B_{i,5}B_{i,6} \quad (B_{i,7}B_{i,8})$. Here, either of the last two B frames from the previous GOP, $B_{i-1,7}B_{i-1,8}$, is included for consideration of possible transmission with the ith GOP provided that the following two conditions are met: 1) $i \geq 2$ and its previous reference frame, i.e., the last P frame from the GOP, $P_{i-1,3}$, is included for transmission in the previous GOP at stage $i-1$; and 2) it is not selected for discarding in the previous GOP at stage $i-1$. The last two frames in the current GOP are included here for consideration of discarding only. They will be considered for transmission in the next GOP at stage $i+1$, because their future reference frame, the I frame in the next GOP, has to be transmitted before them.

For the frames included in the ith GOP (excluding $B_{i,7}B_{i,8}$), $I_i[B_{i-1,7}B_{i-1,8}]P_{i,1}B_{i,1}B_{i,2}P_{i,2}B_{i,3}B_{i,4} \quad P_{i,3}B_{i,5}B_{i,6}$, we check to see whether they can be transmitted using the greedy schedule in such a manner that all of them can be played back in time at the client. If this is the case, we move to the next stage and consider transmission of the $i+1$th GOP. If this is not the case, then one or more frames must be discarded due to the network bandwidth constraint. Note that no more than 14 frames needed to be discarded at each stage. This is because in the worst case we can always discard all the frames in the current (i.e., the ith) GOP including $B_{i,7}B_{i,8}$ and move to the next stage. Now the question is to find the minimum number of frames, m_i^*, $1 \leq m_i^* \leq 14$, that must be discarded at stage i so that some of the frames in the ith GOP (excluding $B_{i,7}B_{i,8}$) can be transmitted using the greedy schedule in such a manner that these frames can be played back in time at the client (we refer to this condition as the *maximum playback constraint*). Furthermore, the m_i^* discarded frames are chosen in such a manner that the buffer gain at the end of the ith GOP is maximized (we refer to this condition as the *maximum buffer gain criterion*).

To find m_i^*, we start with $m_i^* = 1$. Among all the frames that have been included for transmission from the previous stages and the frames in the current GOP, we check to see whether we can find a single frame to discard so that the maximum playback constraint can be satisfied. (Frames that are eligible to discard as a single frame include all the B frames and those "isolated" P or I frames whose dependent B and P frames have already been discarded in the previous stages.) If the maximum playback constraint *can* be satisfied by discarding a single frame, we choose to discard the one among all the eligible single frames that meets the maximum buffer gain criterion. If the maximum playback constraint *cannot* be satisfied by discarding a single frame, we proceed to consider $m_i^* = 2$. More generally, consider the case $m_i^* = m$, $1 \leq m \leq 14$. Among all the frames that have been included for transmission from the previous stages and the frames in the current GOP, we check to see whether we can choose exactly m frames (including the dependent frames of any chosen frame) to discard so that the maximum playback constraint can be satisfied. If this cannot be done, we proceed to consider $m_i^* = m + 1$. If this can be done, we choose those m frames to discard such that the maximum buffer gain criterion be met. Note that this procedure stops when $m_i^* = 14$, as we can always discard the current GOP to meet the maximum playback constraint. Thus we can always find 14 frames to discard such that the maximum buffer gain criterion is met.

Based on a similar argument as used in Theorem 3, it can be shown that the modified MINFD algorithm described above produces a feasible transmission schedule that minimizes the total number of frames discarded. The time complexity of the algorithm is polynomial in N, the number of GOPs in an MPEG video stream. However, the exponent of the polynomial is in the order of the GOP size.

The modified MINFD algorithm for MPEG video minimizes the total number of frames discarded. A variation of this MINFD algorithm can also be devised using a different metric — one that, first of all, minimizes the total number of I frames discarded, then the total number of P frames discarded, and then the total number of B frames discarded. In other words, a P frame is discarded only if it cannot be included by

[3]The algorithm described below does not require the video sequence to have a fixed GOP pattern.

discarding any number of B frames. Similarly, an I frame is discarded only if it cannot be included by discarding any number of B and P frames. Using this metric, we can ensure that each GOP has at least its I frame preserved whenever possible, thereby ensuring certain level of perceived video quality at the client.

An Optimal Selective Frame Discard Algorithm

In this section we describe a general optimal algorithm for selective frame discard, which can be used to determine an optimal feasible set for a given cost function. For simplicity of exposition, the algorithm is described in detail for video streams without inter-frame dependence such as JPEG video streams. Modifications needed to handle inter-frame dependence for encoding schemes such as MPEG are presented at the end of the section.

Recall that a feasible set is considered *optimal* with respect to a given cost function, if it has the least cost among all feasible sets in $\mathscr{SFD}$. We describe an algorithm, referred to as OPTFD, which constructs an optimal feasible set for any given cost function. The algorithm uses the standard dynamic programming technique to find the optimal schedule by formulating the problem as a shortest cost path problem.

The algorithm proceeds in stages. We refer to the collection of transmission sets constructed at each stage as the *system*. At each stage, the system could be in one of several possible states which are determined by the buffer occupancy, resulting from a particular transmission set, and the state information required by the cost function. Each of these states can lead to new states in the next stage as shown in Figure 6. Each transition incurs a certain amount of cost, which depends on the cost function being used. Forward search in dynamic programming can be used [10,23] to find the shortest cost path from stage 0 to N, and thus determine the optimal feasible set. Table 2 summarizes the notation used in this section. For simplicity, we drop the reference to a specific transmission set S from the notation.

At each stage $i, i = 0, 1, \quad , N$, a tuple (B_i, W_i) represents a *state* with which a feasible set S_i is associated, where B_i is the buffer occupancy of the system at the state, and W_i, represents the information needed for computing the transition cost from state (B_i, W_i) to a state (B_{i+1}, W_{i+1}) at the next stage. W_i

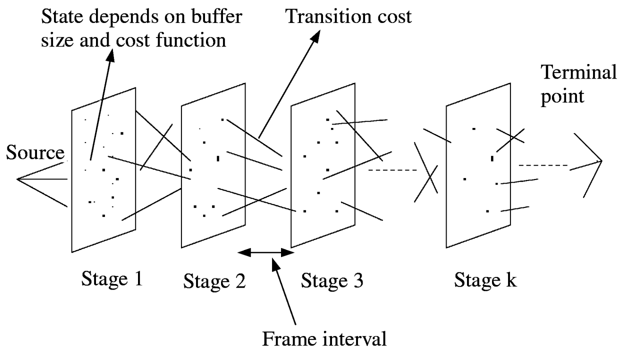


Figure 6. Transitions between the states for the optimal algorithm OPTFD.

depends on the specifics of the cost function. For example, for a simple cost function which assigns a unit cost to each discarded frame, $W_i = \emptyset$. This is because under this simple cost function, the transition cost is determined solely by whether the current frame is discarded or not. For a general cost function, W_i may be determined by a subset of the associated set S_i. Clearly, two different sets S_i and S_i' may be mapped to the same state at stage i. This is the case if $B(S_i) = B_i(S_i')$ and $W_i(S_i) = W_i(S_i')$. Let G_i denote the set of all states at stage i. The set of states G_i can be regarded as a two dimensional mesh, where each point corresponds to a state determined by B_i and W_i. The cost of state (B_i, W_i) is denoted by $C(B_i, W_i)$.

For a given state, (B_i, W_i), $N_{i+1}(B_i, W_i)$ denotes the set of states that can be attained at stage $i+1$ with no constraint violation, namely, neither buffer overflow nor buffer underflow occurs. The set of states G_{i+1} at state $i+1$ can be stated as $G_{i+1} = \cup_{(B_i, W_i) \in G_i} N_{i+1}(B_i, W_i)$. Given the buffer occupancy B_i at stage i, the buffer overflow can be avoided by controlling the amount of data transmitted during the next time slot. Given that the greedy schedule is used, then $a_{i+1} = \min\{C, B - B_i\}$. The buffer underflow can be avoided by checking whether $B_i + a_{i+1} < f_{i+1}$. If this is the case, then frame $i+1$ is not included in the transmission set in stage $i+1$. In this case, the buffer occupancy at the next stage $B_{i+1} = B_i + a_{i+1}$ and the feasible transmission set at next stage $S_{i+1} = S_i$. On the other hand, if $B_i + a_{i+1} \geq f_{i+1}$, then we have the option of either including frame $i+1$ in the transmission or that of excluding it. If i is included, i.e., $S_{i+1} = S_i \cup \{i\}$, then $B_{i+1} = B_i + a_{i+1} - f_{i+1}$. If i is not included, i.e., $S_{i+1} = S_i$, then $B_{i+1} = B_i + a_{i+1}$.

Table 2. Notation for the optimal algorithm OPTFD

(B_i, W_i)	:	a state at stage i where B_i is the buffer occupancy and W_i is the cost function dependent information
J_i	:	shorthand notation for sate (B_i, W_i)
S_i	:	a feasible set associated with (B_i, W_i)
G_i	:	set of states at stage i
N_{i+1} (B_i, W_i)	:	set of states at stage $i+1$ that can be reached from (B_i, W_i) with no constraint violation
$C_i(B_i, W_i)$	:	cost incurred in discarding the frames not in S_i at state J_i
$C^i_{J_i->J_{i+1}}$	:	transition cost from (B_i, W_i) to (B_{i+1}, W_{i+1})

From the above discussion, we see that for any state (B_i, W_i), B_{i+1} can have at most two different values depending on whether frame $i+1$ is included in S_{i+1} or not. W_{i+1} can be determined from W_i depending on the state information required by the cost function. Each possible combination of B_{i+1} and W_{i+1} that can be reached from (B_i, W_i) determines a new state (B_{i+1}, W_{i+1}) that belongs to $N_{i+1}(B_i, W_i)$. For simplicity, we use J_i as a shorthand notation for state (B_i, W_i), and similarly, J_{i+1} for (B_{i+1}, W_{i+1}). Let $C^i_{J_i->J_{i+1}}$ denote the transition cost from state J_i to J_{i+1}. $C^i_{J_i->J_{i+1}}$ is determined by the cost function depending on whether S_{i+1} includes the $i+1^{th}$ frame or not. If it is included in the set, then the transition cost is set to 0. Otherwise, the cost function is used to compute the transition cost based on the state information. Since more than one state at stage i can lead to the same state at stage $i+1$, the cost of state (B_{i+1}, W_{i+1}), $C_{i+1}(B_{i+1}, W_{i+1})$, is the smallest of the sum $C_i(B_i, W_i) + C^i_{J_i->J_{i+1}}$ over all states (B_i, W_i) such that (B_{i+1}, W_{i+1}) can be reached from (B_i, W_i), i.e., $(B_{i+1}, W_{i+1}) \in N_{i+1}(B_i, W_i)$. To put it formally, we have

$$C_{i+1}(B_{i+1}, W_{i+1}) = \min_{(B_i,W_i)\in G_i} \times \left\{ C_i(B_i, W_i) + C^i_{J_i->J_{i+1}} \quad (B_{i+1}, W_{i+1}) \in N_{i+1}(B_i, W_i) \right\}. \tag{9}$$

The problem has now been reduced to a shortest path problem. The objective is to find a feasible set which corresponds to a minimal cost path from the initial state to a state in stage N which has the smallest cost. We can use forward search in dynamic programming to solve the problem. The algorithm is described in pseudo-code in Figure 7. In line 2, we start with the initial state $(B_0, W_0) = (0, \emptyset)$ whose cost is 0. Lines 4–6 of the algorithm determine the set of states G_{i+1} that can be reached at stage $i+1$ from states G_i at stage i, and compute the associated cost for each state $J_{i+1} \in G_{i+1}$ using (9). Finally at stage N, a state with the lowest cost is found (line 7), and a set S_N associated with this state is an optimal set S^* we are looking for (line 8).

The computational complexity of the algorithm depends on the number of stages, and the number of the states at each stage. Let W be the largest size of W_i needed for computing the cost of state J_i at any stage i. Then the complexity of the OPTFD algorithm is $O(BWN)$, since there are N stages and there can be a maximum of BW states at each stage. In the worst possible case we may need to maintain 2^N possible states for each stage, since the cost function dependent state information may require the current set S_i. However, for a realistic cost function like the one introduced in the next section, this state will be much smaller. For an example, consider a simple scenario where the cost of discarding a frame is independent of the cost of discarding any other frame. Then there is no need to maintain W_i as the state can be uniquely determined by B_i. In that case, the complexity of the algorithm is $O(NB)$. In addition, in order to obtain an optimal set S^*, a feasible set S_i associated with each state J_i has to be maintained at each stage. This requires a large amount of memory.

The algorithm as described above applies to video streams with no inter-frame dependencies, e.g., JPEG encoded video streams. To apply the algorithm to an encoding scheme such as MPEG, we need to modify the algorithm to take inter-frame dependence into

```
1.  PROCEDURE OPTFD(C, B)
2.      $G_0 = \{(B_0, W_0)\}, B_0 = 0, W_0 = \emptyset, C_0(B_0, W_0) = 0$
3.      For $i = 0$ to $N-1$ do
4.          Compute $G_{i+1}$ from states $J_i = (B_i, W_i) \in G_i$
5.          For each $J_{i+1} = (B_{i+1}, W_{i+1}) \in G_{i+1}$
6.              compute the cost of $(B_{i+1}, W_{i+1})$ using (9)
7.      At stage $N$, find a state $J_N^*$ with the minimum cost
8.      Output a set associated with the state $J_N^*$
9.  END PROCEDURE
```

Figure 7. The optimal selective frame discard algorithm (OPTFD) for a given cost function.

consideration during the transition from states in the current stage to those in the next stage. As previously, we use the MPEG encoding scheme as an example to illustrate how this can be done. Assume that the frames of a MPEG video stream are transmitted in the decoding order. To account for the dependence among I, P and B frames, we use an additional state variable X_i. X_i can take three possible values: $X_i = 0$ means that the most recent (in the decoding order) I or P frame before frame i has been discarded; $X_i = 1$ means that the most recent I or P frame before frame i has been transmitted but the second most recent I or P frame before frame i has been discarded; and $X_i = 2$ means that the two most recent I or P frames before frame i have both been transmitted. In addition to the playback deadline constraint and the buffer constraint, whether the current (i.e. the ith) frame can be transmitted is also determined by the inter-frame dependency constraint represented by the state variable X_i as follows: the current frame must be discarded if it is a B frame and $X_i < 2$ or if it is a P frame and $X_i = 0$. The value of the state variable at the $i+1$th stage, X_{i+1}, depends only the value of X_i and the type and transmission status of the current frame. If the current frame is either an *I* frame or a *P* frame, the state transition from X_i to X_{i+1} is depicted in Figure 8. Namely, if the current *I* or *P* frame is discarded, then $X_{i+1} = 0$; otherwise $X_{i+1} = X_i + 1$ if $X_i = 0$ or 1, and $X_{i+1} = X_i$ if $X_i = 2$. If the current frame is a *B* frame, then $X_{i+1} = X_i$ no matter whether the current *B* frame is discarded or not. With the help of this state variable, the set of all states at stage $i+1$, G_{i+1}, can be computed from states $J_i = (B_{i+1}, W_{i+1}) \in G_i$ accordingly (see line 4 in Figure 7). Therefore, the same dynamic programming approach can be used to devise an optimal selective frame discard algorithm for MPEG video, given an appropriate cost function.

Heuristic Selective Frame Discard Algorithms for JPEG

As mentioned earlier the complexity of the optimal selective frame discard algorithm, OPTFD, introduced

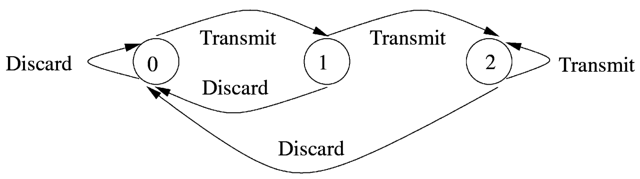


Figure 8. Transitions from X_i to X_{i+1} when current frame is I or P.

earlier is $O(BNW)$. For large values of B and N, this can result in very high complexity. In this section we design a set of efficient heuristic algorithms that aim at minimizing the cost associated with the discarded frames. Most of these heuristics are designed based on the MINFD algorithm and hence have a low computational complexity.

Recall that the MINFD algorithm finds the minimum number of frames that must be discarded for a feasible schedule. However it may tend to discard consecutive frames if large frames are clustered together. Hence the *playback discontinuity* at the client may be very high. In order to provide a measure of this *playback discontinuity*, we define a cost function, $\phi(S)$, that takes two aspects of playback discontinuity into consideration: the length of a sequence of consecutive discarded frames and the spacing or distance between two adjacent but non-consecutive discarded frames.

The cost function $\phi(S)$ assigns a cost c_i to a discarded frame i depending on whether it belongs to a sequence of consecutive discarded frames or not. If frame i belongs to a sequence of consecutive discarded frames, then the cost c_i is defined to be l_i, if frame i is the l_i^{th} consecutively discarded frame in the sequence. Otherwise, the cost c_i is defined based on its distance d_i to the previous discarded frame and given by the formula $c_i = 1 + 1/\sqrt{d_i}$. Therefore, for a set $S \in \mathcal{N}$, the total cost of S is $\phi(S) = \sum_{j \in \mathcal{N} \setminus S} c_j$.

Obviously there are many other ways to define a cost function. We believe that the two aspects of playback discontinuity considered by $\phi(S)$, namely the cost due to consecutive discard and that due to spacing between discarded frames, are important measures of the perceived quality. Any other cost function should reflect these two aspects of playback discontinuity in one way or another. More study[4] is needed in this area to come up with a more realistic cost function based on perceptual quality of video playback [29]. In the rest of this section we will describe a set of heuristic algorithms based on the cost function $\phi(S)$ defined above and results of performance evaluation are then

[4]Although a lot of attention has been devoted to development of accurate perceptual models for video encoding [28], unfortunately most these models are not directly applicable to our study here. We need a relative frame-based perrceptual model to design a meaningful cost function as a basis for selective frame discarding.

```
1.  PROCEDURE JITFD(N)
2.    For $i = 1$ to $N$
3.      Increment the buffer by $\hat{a}_i$
4.      If buffer occupancy $> f_i$
5.        Decrement the buffer by $f_i$ and display frame $i$
6.      Else
7.        Discard frame $i$
8.  END PROCEDURE
```

Figure 9. The JITFD selective frame discard algorithm.

presented. Our algorithms can be easily modified to incorporate the specifics of other cost functions.

Heuristic algorithms

The heuristic algorithms aim at finding a low cost feasible set S by taking either the cost of discarding a frame directly into consideration or indirectly. They differ in the criteria used in selecting a frame to discard. All the heuristics use the greedy schedule to determine the amount of data to be transmitted in each time slot.

As a simple baseline algorithm, we first introduce the *just-in-time* selective frame discard heuristic, *JITFD*. JITFD is perhaps the simplest and most intuitive selective frame discard approach. It always discards the *current* frame whenever its playback deadline cannot be met, irrespective of its cost. The algorithm is shown in Figure 9. At each time i, the buffer is increased by $a_i = \min(B - B_{i-1}, C)$ (line 3), as per the greedy transmission schedule. If the buffer occupancy is smaller than the size of the current frame i, i.e. $B_i + a_i < f_i$, the frame is discarded as in lines 4–7. The computational complexity of this algorithm is linear in N.

The *distance* based selective frame discard algorithm, DISTD(λ), uses a parameter λ to indirectly control the cost of discarded frames. The basic structure of the algorithm is the same as the MINFD algorithm. For any given $\lambda \geq 1$, DISTD(λ) attempts to space the discarded frames λ distance apart by incorporating a distance based priority in selecting a frame to discard. The procedure to select a frame to discard is presented in Figure 10. At each time i, if the playback deadline of frame i is violated, the procedure is invoked. The procedure finds a frame, k, with highest priority p_k, among all frames selected for transmission since the last buffer full point i_0. Here the priority p_k of a frame is defined based on its distance d_k from the previously discarded frame: $p_k = \min\{\lambda, d_k\}$ (line 2). Hence all frames with a distance at least λ are treated with the same priority. Frames are considered for discarding in the order of decreasing priority. Frames with highest priority, namely, $p_j = \lambda$, are considered first. If such a frame cannot be found, all frames with distance λ -1 are considered, and so forth. Among the frames with the same priority, the frame with the largest gain ^i_k is chosen (line 8). Finally, the selected frame k is chosen for discarding only if its gain ^i_k is bigger than the size of the current frame, f_i (this criterion is not shown in Figure 10). Otherwise, the current frame i is discarded.

The *minimum cost* based selective frame discard algorithm, *MINCD*, takes the cost of discarding a frame

```
1.   PROCEDURE DISTD_Select (i, λ)
2.     Set $k = i$; $p_k = \min(d_k, \lambda)$
3.     For $j = i_0 + 1$ to $i - 1$
4.       Do not consider $j$ if $j \notin S$
5.       $p_j = \min(d_j, \lambda)$
6.       If $p_j > p_k$
7.         Set $k = j$
8.       Else If $p_j = p_k$ and $\Delta^i_j > \Delta^i_k$
9.         Set $k = j$
10.  END PROCEDURE
```

Figure 10. Procedure to select the frame to discard for DISTD.

```
1.  PROCEDURE MINCD_Select(i)
2.    Set $k = i$
3.    For $j = i_0 + 1$ to i–1
4.      Do not consider $j$ if $j \notin S_i$
5.      If $c^i_j < c^i_k$
6.        Set $k = j$
7.      Else If $c^i_j = c^i_k$ and $\Delta^i_j > \Delta^i_k$
8.        Set $k = j$
9.  END PROCEDURE
```

Figure 11. Procedure to select the frame to discard for MINCD.

```
1.  PROCEDURE MCMGD_Select(i)
2.    Set $k = i$
3.    For $j = i_0 + 1$ to $i - 1$
4.      Do not consider $j$ if $j \notin S_i$
5.      If $\Delta^i_j / c^i_j > \Delta^i_k / c^i_k$
6.        Set $k = j$
7.  END PROCEDURE
```

Figure 12. Procedure to select the frame to discard for MCMGD.

Table 3. Characteristics of JPEG video traces

Title	Length (min)	No. of frames	Ave. rate (Mbps)	Peak rate (unsmoothed)	Peak rate (smoothed)
Sleepless in Seattle	101	181457	2.28	3.99	3.30
Beauty and Beast	80	143442	3.04	7.29	6.54
Jurassic Park	112	220061	2.73	5.73	4.78

directly into consideration. The procedure for selecting the frame to discard is given in Figure 11. At time i, if the playback deadline of frame i is violated, a frame k with lowest incurred cost c^i_k is chosen for discarding. Let S_{i-1} be the feasible set constructed at time $i-1$. The incurred cost c^i_k is defined to be the cost incurred if frame k is discarded at time i, i.e., $c^i_k = \phi(S_{i-1}) - \phi[(S_{i-1} \cup \{i\}) \backslash \{k\}]$. As shown in lines 3–6, a frame with the smallest incurred cost is chosen for discarding. If two frames have the same incurred cost, the one that yields larger gain ${}^{i}_{j}$ is chosen (lines 7–8).

The last heuristic we consider is the *minimum cost maximum gain* based selective frame discard heuristic, *MCMGD*. In selecting a frame to discard, it takes both the gain ${}^{i}_{j}$ from discarding a frame and the cost c^i_j incurred thereof into consideration. The procedure for selecting the frame to discard is shown in Figure 12. It discards a frame k with the largest gain to the incurred cost ratio, i.e., ${}^{i}_{j}/c^i_j$ (lines 5–6). By discarding frames with the largest gain to cost ratio, the MCMGD heuristic uses in effect the steepest gradient search for an optimal solution.

The computational complexity of the DISTD, MINCD and MCMGD heuristics is $O(N^2)$. This is much smaller than the computational complexity of the optimal algorithm OPTFD.

Performance evaluation

In this section, we evaluate the performance of the heuristic selective frame discard algorithms using JPEG video traces. For given bandwidth and client buffer size constraints, the number of frames discarded and the cost incurred by these algorithms are compared. The cost is computed using the heuristic cost function defined earlier. The impact of each constraint on the performance of these algorithms is also studied by varying one constraint while keeping the other constraint fixed. We present the results for three representative traces, "Sleepless in Seattle", "Beauty and the Beast" and "Jurassic Park". Table 3 lists the characteristics of these traces [31], where among other things, the average rate, the peak rate of the video traces are shown. Also included is the peak rate of the optimal smoothed schedule [4] using a client buffer size of 1 MB and zero startup delay.

Table 4 compares the performance of various selective frame discard algorithms. The rate constraint C in each case is set to the average rate of the video trace, while the client buffer size B is set to 1 MB. As shown in Table 3, the peak rate of the optimal smoothed schedule is considerably higher than the chosen rate constraint. Hence, continuous playback is not possible, forcing the server to discard frames. Consider the performance of the heuristic algorithms when applied to the video trace "Sleepless in Seattle". JITFD discards 10538 frames with a cost of 15720.21. DISTD(2) drops 10272 frames, while DISTD(5) drops 10414 frames, larger than that of DISTD(2). However, the cost of DISTD(5) is 15372.894, lower than that of DISTD(2), which is 15696.250. This is due to the fact the discarded frames in DISTD(5) are more distributed than those of DISTD(2), incurring a lower cost despite a larger

Table 4. Comparison of various selective frame discard algorithms for JPEG

Selective frame discard algo	Sleepless in Seattle		Beauty and the Beast		Jurassic Park	
	Discards	Cost	Discards	Cost	Discards	Cost
JITFD	10538	15720.21	8878	13355.89	13457	20269.55
DISTD (2)	10272	15696.25	8602	13370.69	13141	20262.52
DISTD (5)	10414	15372.89	8692	13196.46	13294	19909.13
MINCD	10473	15332.03	8742	13170.24	13384	19845.25
MCMGD	10455	15246.31	8712	13130.756	13342	19746.68
MINFD	9907	128797.77	8183	106951.31	12516	148921.86

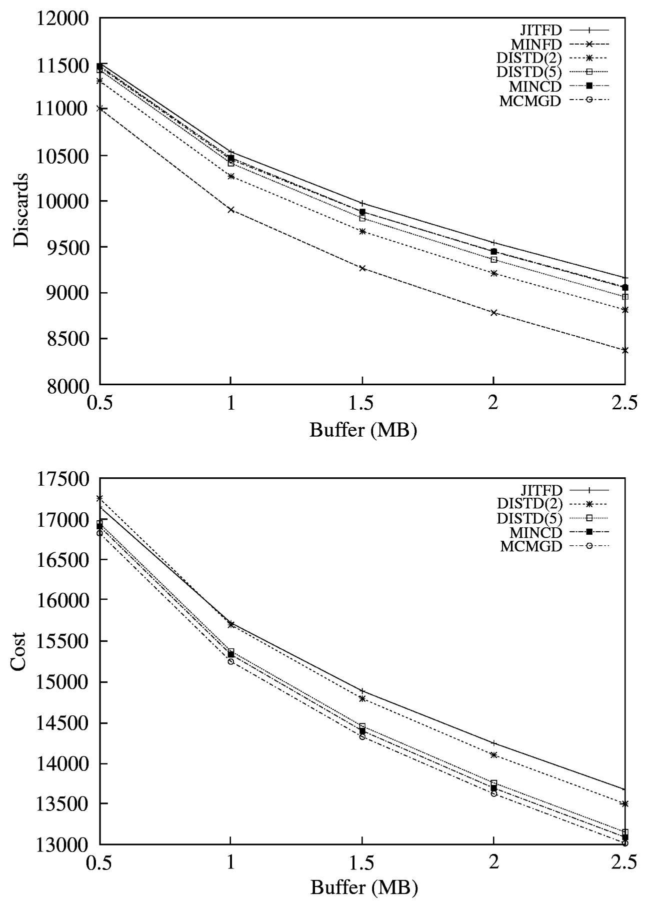


Figure 13. Performance under varying buffer sizes with C fixed at 2.28 Mbps for "Sleepless in Seattle". (a) Buffer size vs number of discarded frames; (b) buffer size vs cost.

number of discarded frames. For the same trace, MINCD discards 10473 frames with a cost of 15332.03, and MCMGD incurs a cost of 15246.31 by discarding 10455 frames. All the heuristic discard schemes that take cost into consideration incur less cost than JITFD does. Among them, MCMGD performs best, as expected. We also ran the optimal algorithm[5], OPTFD, on this trace. It discarded 10455 frames with a cost of 15245.8. We see that the results produced by MCMGD are near optimal. It is also worth pointing out that MINFD indeed gives the lowest number of discards. However, the incurred cost is quite high as it tends to discard consecutive large frames. Clearly, there is a trade-off between reducing the total number of discarded frames and distributing discarded frames in a video stream.

We now study the impact of varying buffer size while fixing the rate constraint on the performance of the selective frame discard algorithms. Figure 13 shows the number of discarded frames as well as the incurred cost as a function of buffer size for the trace "Sleepless in Seattle". The bandwidth C is fixed at 2.28 Mbps, and the client buffer size B is increased from 0.5 MB to 2.5 MB. It can be seen that all the other four heuristic algorithms perform better than JITFD. The difference in performance among the heuristics widens as the buffer size increases. This phenomenon can be explained as follows. Recall that frames which come before a buffer full point

[5]In order to speed up the execution of OPTFD and reduce the memory space, a branch and bound strategy is used to control the number of states by pruning states which are unlikely to lead to an optimal cost. Hence the results produced may not be completely accurate, but we believe they give a good approximation to the true optimal values.

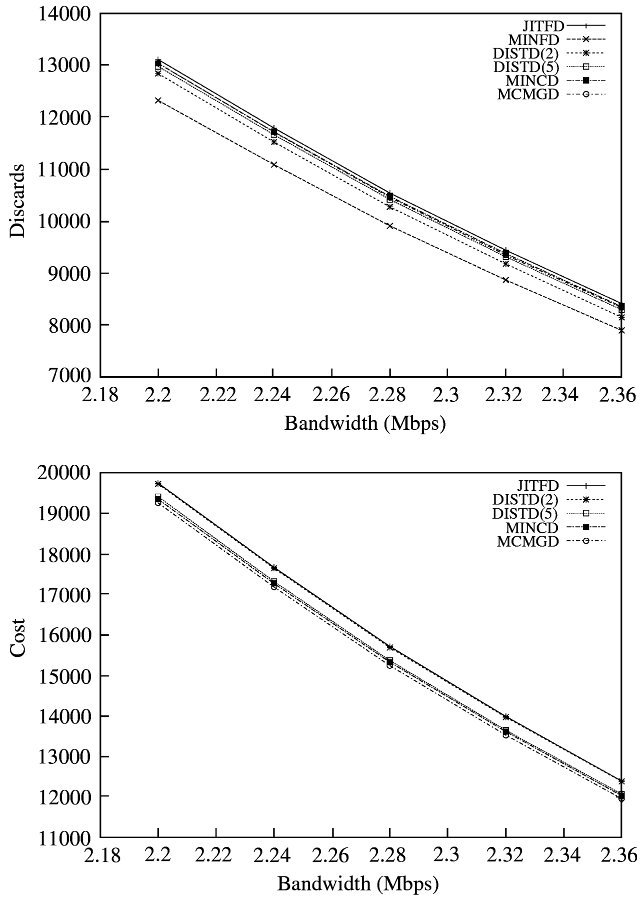


Figure 14. Performance under varying bandwidth with B fixed at 1 MB for "Sleepless in Seattle". (a) Bandwidth vs number of discarded frames; (b) bandwidth vs cost.

Table 5. Characteristics of *Starwars* MPEG video trance

Title	Length (min)	I frames	P frames	B frames	Total frame	Avg. Rate (Mbps)	Peak rate (Mbps)
Starwars	121	14505	43514	116036	174055	0.374	4.446

are not considered for discarding for a deadline violation after the buffer full point. Hence, with increased buffer size, the number of frames from which a frame can be selected for discarding increases. It therefore enhances the effectiveness of the selection criteria used in the heuristics such as MINCD and MCMGD. Among all the heuristics, it is quite evident that MCMGD performs best at all buffer sizes.

Figure 14 shows the impact of bandwidth variation for the trace "Sleepless in Seattle". The bandwidth is varied from 2.18 Mbps to 2.36 Mbps with the client buffer size fixed at 1 MB. As the bandwidth increases, the difference in performance between the JITFD and the other four heuristic algorithms narrows slightly. This is because at a higher bandwidth, the playback deadline of fewer frames are violated. As a result, discarded frames are more likely to be distributed and the advantage of more sophisticated heuristics is less pronounced. The MCMGD algorithm still has the best performance across the bandwidth range.

We have run the heuristic algorithms on other JPEG traces and the results obtained are very similar. We conclude that the proposed heuristic algorithms work well in improving the perceived quality as measured by the proposed cost function. Among them, the MCMGD heuristic has the best performance.

Heuristic Selective Frame Discard Algorithms for MPEG

In this section, we extend the heuristic algorithms developed for JPEG encoded video to handle inter-frame dependencies for MPEG encoded video. The evaluation based on MPEG video traces is then presented.

The following modifications are made to the selective frame discard heuristics in order to take the MPEG inter-frame dependencies into account. If the current frame is a P or B frame, it is discarded if its previous reference frame is not included in the transmission schedule. In addition, a B frame is also discarded if its future reference frame cannot be transmitted before the playback deadline of the current B frame.

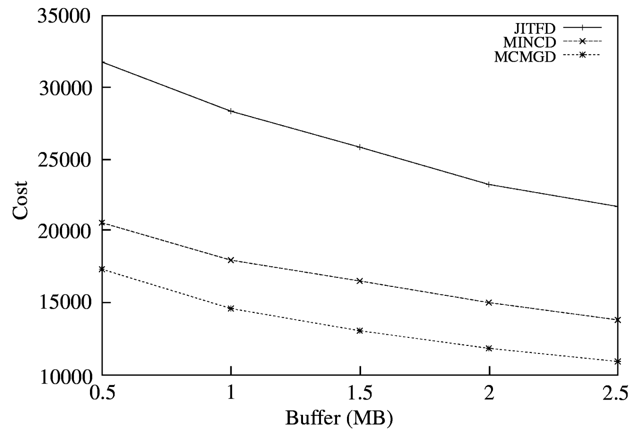
The above modifications are the only changes needed for the *just-in-time* (JITFD) heuristic. For the *minimum cost* (MINCD) and *min-cost-max-gain* (MCMGD) heuristics, an additional modification is incorporated to attach a relative importance to I, P and B frames. In deciding a frame to discard, frames are considered in the following order of relative importance: B, P_3, P_2, P_1, I. In other words, eligible frames of types B are always considered first. If no eligible frames of type B are left, eligible frames of type P_3 are considered, and so forth. As described in the last section, MINCD discards a frame with lowest incremental cost, while MCMGD chooses a frame with largest ratio of gain and incremental cost. Similar extensions can also be incorporated in the distance-based (DISTD) heuristic, where the distance between frames is now defined for each type of frames. For clarity of exposition, we do not include the DISTD heuristic in the following performance evaluation.

The statistics of the MPEG trace is listed in Table 5. The performance of the JITFD, MINCD and MCMGD

Table 6. Comparison of the various selective frame discard algorithms for MPEG

Selective frame discard algo	Discarded frames				Cost
	I	P	B	Total	
JITFG	210	827	5924	6961	31761.86
MINCD	1	18	11948	11967	20597.93
MCMGD	1	18	9860	9879	17364.47

heuristics is evaluated and compared in Table 6 using the *Star Wars* MPEG video trace. The rate constraint C is set to 0.4 Mbps and the client buffer size B is fixed at 0.5 MB. From Table 6, we see that JITFD performs significantly worse than MINCD and MCMGD, unlike the case of JPEG. This is due to the fact that JITFD does not take into account the frame dependencies and the relative importance of frame types. Hence it is prone to discard a large number of I frames, incurring a much higher cost, as compared to that of MINCD and MCMGD. Whereas MINCD and MCMGD attempt to minimize the chance of discarding I and P frames by considering frame types in the order of their relative importance. MCMGD discards fewer B frames than MINCD, while the same number of I and P frames are discarded by both heuristics. MCMG performs better than MINCD, because it takes both cost and gain into consideration.



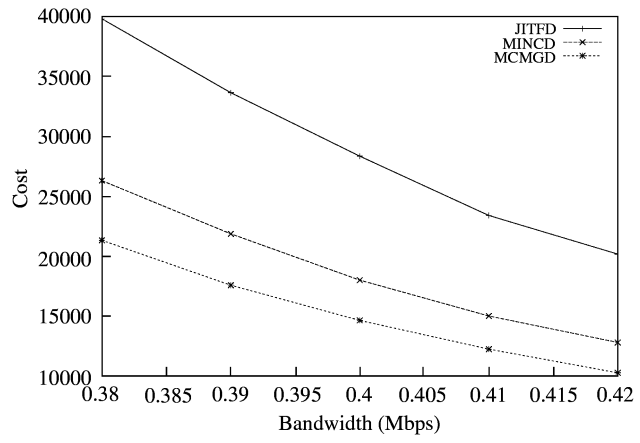


Figure 15. performance under (a) varying buffer sizes, buffer size vs cost (C = 0.4 Mbps); (b) varying bandwidth for "Starwars". bandwidth vs cost (B = 0.5 MB).

The impact of varying client buffer size and network bandwidth on the performance of these heuristics is shown in Figure 15 (a) and (b) respectively. In both cases, as the client buffer size or network bandwidth increases, the cost for each heuristic decreases. This is because with larger buffer or higher network bandwidth, the likelihood of discarding I or P frames is reduced, thereby reducing the overall cost. We have observed similar results for other MPEG traces.

Conclusions

In this paper, we have developed various selective frame discard algorithms for stored video delivery across a network where both the network bandwidth and the client buffer capacity are limited. We began by formulating the problem of optimal selective frame discard using the notion of a cost function. The cost function captures the perceived video quality at the client. Given network bandwidth and client buffer constraints, we developed an $O(N \log N)$ algorithm to find the minimum number of frames that must be discarded in order to meet these constraints. The correctness of the algorithm is also formally established. We presented a dynamic programming algorithm for solving the optimal selective frame discard problem. Since the computational complexity of the optimal algorithm is prohibitively high in general, we also developed several efficient heuristic algorithms for selective frame discard. These algorithms are evaluated using JPEG and MPEG video traces. We found that the *minimum cost maximum gain* algorithm performs best for both JPEG and MPEG encoded video.

In this paper, we have considered a network model where the network bandwidth is fixed and is known a priori, as is the case in a network with CBR service. We can easily extend our work to the case where the network bandwidth can vary, but the bandwidth variation is known to the server beforehand. To address the case where the network bandwidth is unknown, we are currently working on adaptive selective frame discard schemes using feedback-based bandwidth estimation mechanisms. Initial work in this direction is reported in [30]. We are currently conducting experiments to evaluate our schemes across a real network. Evaluation of server selective frame discard algorithms based on the actual QoS perceived by clients will then be carried out. In addition, we are also applying the MINFD algorithm in the context of layered video to address the problem of optimal selective layer discarding.

Acknowledgements

This work was supported in part by a University of Minnesota Graduate School Grant-in-Aid grant, NSF CAREER Award grant NCR-9734428, NSF grant ANIR-9903228, and by US Department of Energy grant DE-ACO4-94-AL85000. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation or US Department of Energy. We are grateful for the generous support of these funding agencies. We would also like to thank the anonymous reviewers for their insightful comments and suggestions.

References

1. Garrett, M. & Willinger, W. (1994) Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. *Proc. ACM SIGCOMM*, *London*, *Aug. 1994*, pp. 269–280.
2. Krunz, M., & Tripathi, S.K. (1997) On the Characteristics of VBR MPEG Streams. In *Proc. ACM SIGMETRICS*, *Seattle*, *U.S.A.*, *June 1997*, pp. 192–202.
3. Lakshman, T.V., Ortega, A. & Reibman, A.R. (1998) Variable Bit-rate (VBR) Video: Tradeoffs and Potentials. *Proceedings of the IEEE* **86**: 952–973.
4. Salehi, J.D., Zhang, Z.-L., Kurose, J.F. & Towsley, D. (1996). Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. In *Proc. ACM SIG-METRICS, Philadelphia PA, May 1996*, pp. 222–231.
5. Zhang, J. & Hui, J.Y. (1997) Traffic Characteristics and Smoothness Criteria in VBR Video Traffic Smoothing. *IEEE International Conference on Multimedia Computing and Systems*, *Ottowa*, *Ontario*, *Canada*, *June 1997*.
6. Feng, W. & Rexford, J. (1997) A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video. In *Proc. IEEE INFOCOM*, *Kobe*, *Japan*, *April 1997*, pp 58–66.
7. Zhang, Z.-L., Kurose, J., Salehi, J.D. & Towsley, D. (1997) Smoothing, Statistical Multiplexing and Call Admission Control for Stored Video. *IEEE Journal on Selected Areas in Communication, Special Issue on Real-Time Video Services in Multimedia Networks, August 1997*, pp. 1148–1166.
8. McManus, J.M. & Ross, K.W. (1996) Video on Demand over ATM: Constant-rate Transmission and Transport *Proc. IEEE INFOCOM*, *San Francisco*, *U.S.A.*, *March 1996*, pp. 1357–1362.
9. Rexford, J., Sen, S., Dey, J., Feng, W., Kurose, J., Stankovic, J. & Towsley, D. (1997) Online Smoothing of Live, Variable-bit-rate Video. In *Proc. Workshop on Network and Operating System Support for Digital Audio and Video*, *St. Louis, Missouri, U.S.A., May 1997, pp. 249–257.*
10. Jiang, Z. & Kleinrock, L. (1998) A General Optimal Video Smoothing Algorithm. In *Proc. of the IEEE INFOCOM'98 Conference, San Francisco, CA, Mar. 1998*.
11. Reibman, A.R. & Haskell, B.G. (1992) Constraints on Variable Bit-Rate Video for ATM Networks. *IEEE Transactions on Circuits and Systems for Video Technology* **2**: 361–372.
12. Wu-chi Feng (1997) Rate-constrained Bandwidth Smoothing for the Delivery of Stored Video. *SPIE Multimedia Computing and Networking*, *San Jose*, *CA*, *U.S.A.*, *1997*, pp. 316–327.
13. Rexford, J. & Towsley, D. (1997) Smoothing Variable-bit-rate Video in an Internetwork. In *Proc. SPIE Symposium on Voice*, *Video*, *and Data Communications: Multimedia Networks: Security*, *Displays*, *Terminals*, *and Gateways*, *San Jose*, *CA*, *U.S.A.*, *November 1997*.
14. Sahu, S. Zhang, Z.-L., Kurose, J. & Towsley, D. (1997) On the Effiecient Retrieval of VBR Video in a Multimedia Server. In *Proc. IEEE International Conference on Multimedia Computing and Systems'97*, *June 1997, Ottawa*, *Ontario*, *Canada*, pp. 46–53.
15. Sen, S., Towsley, D., Zhang, Z.-L. & Dey, J. (1999) Optimal Multicast Smoothing of Streaming Video over an Internetwork. In *Proc. IEEE INFOCOM'99*, *New York City*, *New York*, *March 1999*.
16. Grossglauser, M., Keshav, S. & Tse, D. (1995) RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. In *Proc. ACM SIGCOMM'95*, *Cambridge, MA*, *U.S.A.*, *Aug 1995*, pp. 219–230.
17. Ding, W. (1997) Joint Encoder and Channel Rate Control of VBR Video over ATM. *IEEE Trans. Circuit Syst. Video Technol.* **7**: 266–278.
18. Duffield, N.G., Ramakrishnan, K.K. & Reibman, A. (1998) SAVE: An Algorithm for Smoothed Adaptive Video over Explicit Rate Networks. In *Proc. of the IEEE INFOCOM'98 Conference*, *San Francisco*, *CA*, *Mar. 1998*.
19. Hsu, C., Ortega, A. & Reibman, A. (1997) Joint Selection of Source and Channel Rate for VBR Video Transmission under ATM Policing Constraints. *IEEE Journal on Selected Areas in Communication* **15**: 1016–1028.
20. Kanakia, H., Mishra, P.P. & Reibman, A. (1993) Packet Video Transport in ATM Networks with Single-bit Feedback. In *Proc. of the Sixth International Workshop on Packet Video*, *Portland*, *Oregon*, *Sept. 1994*. *SIGCOMM'93 Conference*, *San Francisco*, *CA*, *U.S.A.*, *Sept. 1993*.
21. Lakshman, T.V., Mishra, P.P. & Ramakrishnan, K.K. (1997) Transporting Compressed Video over ATM Networks with Explicit Rate Feedback Control. In *Proc. of the IEEE INFOCOM'97 Conference*, *Kobe*, *Japan*, *Apr. 1997*.
22. Luo, W. & El Zarki, M. (1997) Quality control for VBR video over ATM networks, *IEEE Journal on Selected Areas in Communication* **15**: 1029–1039.
23. Servetto, S., Ramachandran, K., Nahrstedt, K. & Ortega, A. (1997) Optimal Segmentation of a VBR Source for its Parallel Transmission over Multiple ATM Connections.

In *Proceedings of the International Conference on Image Processing*, *Washington, DC*, *U.S.A.*, *1997*.

24. Ramanathan, S., Rangan P.V. & Vin, H.M. (1993) Frame-Induced Packet Discarding: An Efficient Strategy for Video Networking, In *Proceedings of the Fourth International Workshop on Network and Operating Systems Support for Digital Video and Audio*, *Lancaster*, *UK*, *November 1993*, pp. 175–187.
25. Rowe, L.A., Patel, K.D., Smith, B.C. & Liu, K. (1994) MPEG Video in Software: Representation, Transmission and Playback. *IS&T/SPIE*, *Symp. on Elec. Imaging Sci. & Tech., San Jose, CA, February 1994*.
26. Rowe, L.A. & Smith, B.C. (1992) A Continuous Media Player. *Proceedings 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, *San Diego*, *CA*, *November 1992*.
27. Zhang, Z.-L., Nelakuditi, S., Aggarwal, R., & Tsang, R.P. (1998) *Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks*. Technical Report, Department of Computer Science, University of Minnesota, July 1998. The paper can be found at http://www.cs.umn.edu/~zhzhang/papers.html.
28. Verscheure, O., Garcia, X., Karlsson, G. & Hubaux, J.P. (1998) User-Oriented QoS in Packet Video Delivery. *IEEE Network* **12**: 12–21.
29. Wijesekera, D. & Srivastava, J. (1995) Quality of Service (QoS) Metrics for Continuous Media. *Multimedia Tools and Applications* **2**: 127–166.
30. Aggarwal, R., Nelakuditi, S. & Zhang, Z.-L. (1998) *Adaptive Stored Video Delivery using Selective Frame Discard across Resource Constrained Networks*. Technical Report, Department of Computer Science, University of Minnesota, *June 1998*.
31. Wu-chi Feng (1996) *Video-on-Demand Services: Efficient Transportation and Decompression of Variable Bit Rate Video*. Ph.D. Thesis, Univ. of Michigan, *April 1996*.