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

Zhi-Li Zhang†, Srihari Nelakuditi†, Rahul Aggarwal†, and Rose P. Tsang‡

† Dept. of Computer Science & Engineering
University of Minnesota
Minneapolis, MN55455
{zhzhang,srihari,raggarwa}@cs.umn.edu

‡ Sandia National Laboratories
PO Box 969, Mail Stop 9011
Livermore, California 94550
rtsang@ca.sandia.gov

Abstract—**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. Since the computational complexity of the optimal algorithm for solving the optimal selective frame discard problem is prohibitively high in general, we also develop several efficient heuristic algorithms for selective frame discard. These algorithms are evaluated using JPEG video traces.**

I. 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 [4], [7]. 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 (see, e.g. [13], [3], [8], [15]) 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 [2], [10], [12]. In reality, however, both network bandwidth and client buffering capacity 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 [5], 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*[1] *(SFD)* at the server which *preemptively* 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 interdependencies, 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. For a given cost function, an optimal algorithm for solving the optimal selective frame discard problem can be designed using dynamic programming. Since the computational complexity of this 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 video traces. Through the performance evaluation, we find that the proposed *minimum cost maximum gain* heuristic algorithm yields near-optimal performance for JPEG encoded

This work was supported in part by a University of Minnesota Graduate School Grant-in-Aid grant, NSF CAREER Award grant NCR-9734428, 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 funding agencies.

[1] 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.

0-7803-5417-6/99/$10.00 ©1999 IEEE.

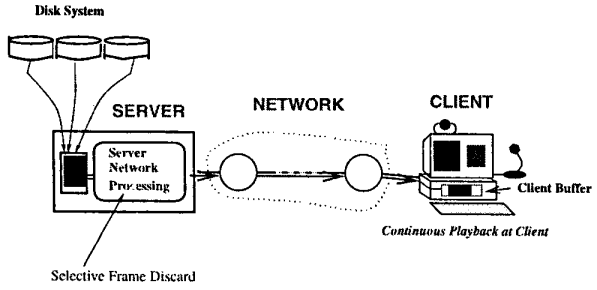


Fig. 1. Overview of the problem setting

video.

Packet discarding schemes which take advantage of application-specific information have been used in many different contexts (see, e.g., [9], [11], [6]). 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.

The rest of this paper is organized as follows. Section II describes the problem setting and formulates the optimal selective frame discard problem. The minimum frame discard algorithm is described and its correctness is proved in Section III. Section IV introduces several efficient selective frame discard heuristics and presents performance evaluation based on JPEG traces. We conclude with Section V.

II. 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 QoS metrics 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 playback deadline 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*.

In a resource constrained system, there may not be sufficient resources to ensure the continuous playback of the video at the client. We consider two specific resource constraints: *rate constraint* and *client buffer constraint*. While the rate constraint 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.

TABLE I
Notation

N	:	length of video in frames.
f_i	:	size of i^{th} frame.
B	:	client buffer capacity for storing unplayed frames.
C	:	network bandwidth.
$\mathcal{N}$	:	set of all frames, i.e, $\{1,\ldots,N\}$
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 $\hat{A}(S)$.

A naive approach at the server may attempt to transmit each frame with no cognizance 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 *preemptively* 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 I summarizes the notation we introduce in this section.

Consider a video stream with N frames. For $i \in \mathcal{N} = \{1,\ldots,N\}$, the size of 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 $S \subseteq \mathcal{N}$, $\mathbf{1}_{\{j\in S\}}$ is the indicator function: $\mathbf{1}_{\{j\in S\}} = 1$ if $j \in S$ and 0 if $j \notin S$. Let $D(S) = \{D_0(S), D_1(S), \ldots, D_N(S)\}$ where $D_i(S) = \sum_{j=0}^{i} f_j \mathbf{1}_{\{j\in S\}}$, and let $U(S) = \{U_0(S), U_1(S), \ldots, U_N(S)\}$ 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, $i = 1,\ldots,N$. In accordance with the notation for $D(S)$ and $U(S)$, the schedule $A(S)$ is denoted by $A(S) = \{A_0(S), A_1(S), \ldots, A_N(S)\}$

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 $i = 0, 1, \ldots, N$, 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 $B_0(S) = 0$.

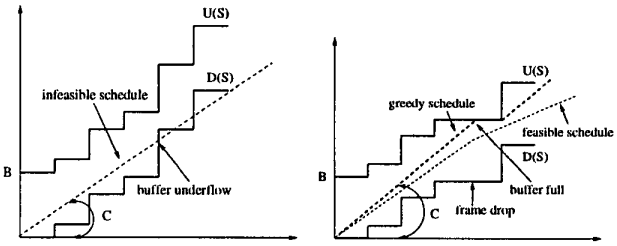
If $B_{i-1}(S) + a_i(S) > B$, the *buffer overflow* occurs at time i. If $B_{i-1}(S) + a_i(S) < f_i$, then *buffer underflow* occurs at time i. Clearly for a feasible schedule, $B_i(S) = B_{i-1}(S) + a_i(S) - f_i \mathbf{1}_{\{i \in S\}}$.

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, $i = 1, \ldots, N$, is given by $\hat{a}_i(S) = min\{B - B_{i-1}(S), C)$, where $B_0(S) = 0$ and $B_i(S) = B_{i-1}(S) + \hat{a}_i(S) - f_i \mathbf{1}_{\{i \in S\}}$. Hence $\hat{A}(S) = \{\hat{A}_0(S), \hat{A}_1(S), \ldots, \hat{A}_N(S)\}$, where $\hat{A}_i(S) = \sum_{j=0}^{i} \hat{a}_j(S)$. 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 $\hat{A}_i \geq D_i(S)$, $i = 0, 1, \ldots, N$. 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 $A_i(S) \leq \hat{A}_i(S)$, $i = 0, 1, \ldots, N$.

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



(a) Infeasible schedule (b) Feasible schedule

Fig. 2. Relation of $D(S)$, $U(S)$ and a server transmission schedule $A(S)$.

the notion of a *cost function*, $\phi(S)$, to quantify the "desirability" of different feasible sets. The cost of a feasible set $\phi(S)$ is the cost associated with the frames that are not part of the set, i.e., the discarded frames. 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 $\phi(S^*) = \min\{\phi(S) : S \in \mathcal{SFD}(C, B)\}$.

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

For a given cost function ϕ, a general optimal algorithm to determine S^* can be designed using dynamic programming. The optimal algorithm proceeds in stages, where stage i corresponds to the ith frame, $i = 1, 2, \ldots, N$. In each stage, a set of appropriate states is maintained. A transition from a state in stage $i-1$ to another state in stage i represents whether frame i is included or discarded at stage i while no constraints are violated. The incurred cost of the transition is computed accordingly using the cost function. The optimal selective frame discard problem can thus be reduced to a shortest path problem and solved using dynamic programming. The computational complexity of the algorithm is $O(NBW)$, where W is the largest size of the states in each stage, which in the worst can be as large as 2^N. Due to the space limitation, we leave the detailed description of this dynamic programming based optimal alogrithm to the extended version of this paper [16].

III. Upper Bound on the Size of Feasible Sets

In this section, we consider the following 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 deadlines under the known network bandwidth (rate) and client buffer constraints?* 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 Section IV. We present an algorithm for solving this problem and establish its correctness. This algorithm is referred to as *minimum frame discard* algorithm, in short MINFD.

Consider a video stream encoded using an intra-frame encod-

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

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

ing scheme such as JPEG. Hence there is no inter-frame dependency among the frames. Following the notation introduced in Section II, 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 MINFD algorithm is presented in pseudo-code in Figure 3. It proceeds in stages, $i = 0, 1, \ldots, N$, and constructs a feasible set $S^{\#}$ iteratively.

At stage 0 (line 2 in the algorithm), we start with $S^{\#} = \emptyset$. 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 3-17), $i = 1, \ldots, N$, we follow the greedy schedule $\hat{A}$, and transmit as much data as possible, namely, $\hat{a}_i = \min\{C, B - B_{i-1}\}$ (lines 4-6). If the buffer is full at this point, set $i_0 = i$. Otherwise (lines 7-17), 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, $1 \le j \le i$, we introduce the notion of *gain* in the buffer occupancy at time i if frame j is discarded. We denote this gain by Δ^i_j; its definition will be given shortly. The frame discarded, say, frame k, is thus the one which yields the largest gain, namely, $\Delta^i_k = \max_{1 \le j \le i} \Delta^i_j$. This is done in lines 9-12.

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 $\hat{A}_j(S^{\#}_{i-1})$ respectively 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 $j = 1, \ldots, i-1$, define

$$\nabla^i_j = \min_{j \le l \le i-1} \{U_l(S^{\#}_{i-1}) - \hat{A}_l(S^{\#}_{i-1})\} \tag{1}$$

∇^i_j 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 Δ^i_j.

$$\Delta^i_j = \begin{cases} f_i, & j = i, \\ \min\{f_j, \nabla^i_j\}, & j = 1, \ldots, i-1. \end{cases} \tag{2}$$

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

$$B_{i-1}(S^{\#}_{i-1} \setminus \{j\}) = B_{i-1}(S^{\#}_{i-1}) + \Delta^i_j. \tag{3}$$

This is shown pictorially in Figure 4 where the two cases: (a) $f_j \le \nabla^i_j$ and (b) $f_j > \nabla^i_j$ are depicted. As a result of discarding frame j, the segment $[j+1, i-1]$ of the new buffer overflow curve $U(S^{\#}_{i-1} \setminus \{j\})$ and underflow curve $D(S^{\#}_{i-1} \setminus \{j\})$ are the original ones ($U(S^{\#}_{i-1})$ and $D(S^{\#}_{i-1})$) shifted f_j amount downwards. Consider the case where $f_j \le \nabla^i_j$. 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 $f_j > \nabla^i_j$, the amount of data transmitted by the greedy schedule during the same interval is only $(i-1-j)C - (f_j - \nabla^i_j)$. This is because the buffer becomes full at some point. Hence the greedy schedule needs to stop transmission for a duration of $(f_j - \nabla^i_j)/C$ time. Thus (3) also holds at time $i-1$.

From (1), $\nabla^i_j = 0$ for any $j \le i_0$. Therefore, discarding any frame before time i_0 will result in zero gain, i.e., $\Delta^i_j = 0$. In other words, *discarding any frame before the last buffer full point will not help meet the playback deadline of frame i.* This

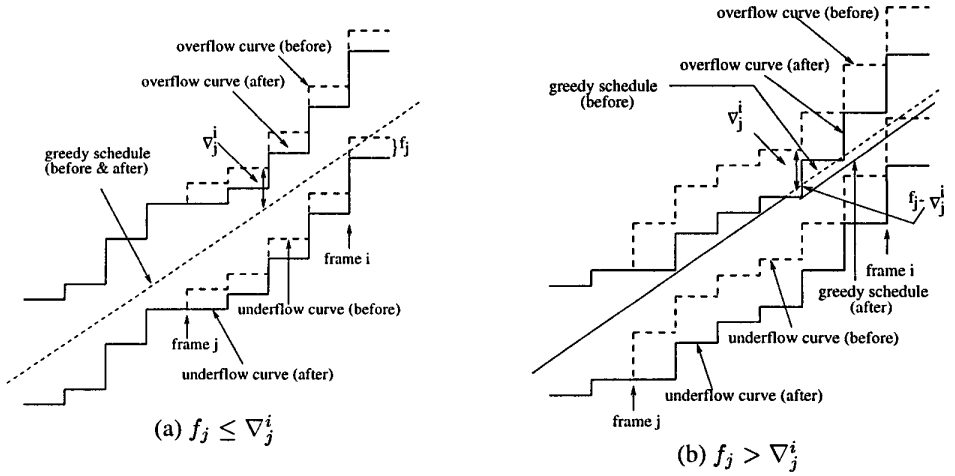


Fig. 4. Effect of discarding a frame j on D, U and $\hat{A}$.

is the reason in line 10 of the algorithm in Figure 3, we only search in the range of $[i_0 + 1, i]$ for a frame to discard. Let k, $i_0 + 1 \le k \le i$ be such that $\Delta_k^i = \max_{i_0+1 \le j \le i} \Delta_j^i$. Hence discarding frame k yields the maximal gain at time i. Denote this maximal gain by $\max \Delta_i$, i.e., $\max \Delta_i = \Delta_k^i$. As $\Delta_i^i = f_i$, we have $\max \Delta_i \ge f_i$. Hence from (3)

$$B_{i-1}(S_{i-1}^{\#} \setminus \{k\}) \ge B_{i-1}(S_{i-1}^{\#}) + f_i.$$

Therefore, if $k \ne i$, 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 $S_i^{\#} := S_{i-1}^{\#} \cup \{i\} \setminus \{k\}$ (lines 13-14), we have

$$B_i(S_i^{\#}) = B_{i-1}(S_{i-1}^{\#}) + C + \max \Delta_i - f_i. \qquad (4)$$

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

In lines 15-17 of the algorithm, the buffer occupancy B_i is updated using (4), and i_0 is set to k^* if discarding k results in a full buffer at time k^*. If the deadline of frame i is met, it is included in $S_i^{\#}$ by setting $S_i^{\#} := S_{i-1}^{\#} \cup \{i\}$ (line 19). The algorithm stops after stage N and outputs the set $S^{\#}$.

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 [16].

Lemma 2: Let S be any feasible set, i.e., $S \in \mathcal{SFD}(C, B)$. Then $|S_i^{\#}| \ge |S \cap \{1, 2, \ldots, i\}|$, where $|\cdot|$ denotes the cardinality of a set. Moreover, for any $j = 1, 2, \ldots, i$, if $|S_i^{\#} \cap \{1, \ldots, j\}| = |S \cap \{1, \ldots, j\}|$, then $B_j(S_i^{\#}) \ge B_j(S)$.

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 meeting 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.

```
1.  PROCEDURE JITFD(N)
2.    For i = 1 to N
3.      Increment the buffer by â_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
```

Fig. 5. The JITFD selective frame discard algorithm.

Theorem 3: Let $S^{\#}$ be the feasible set produced by the MINFD algorithm. Then $|S^{\#}| = \max\{|S| : S \in \mathcal{SFD}(C, B)\}$.

Finally, we remark that by using a clever data structure for maintaining and updating the gain Δ_j^i, we can design an $O(N \log N)$ algorithm to construct $S^{\#}$. We can also modify the MINFD algorithm described in Figure 3 to handle video streams with inter-frame dependencies such as those encoded using the MPEG encoding scheme. The modification needed is fairly elaborate. Due to space limitation, we will not describe it here.

IV. Heuristic Selective Frame Discard Algorithms

As mentioned earlier the computational complexity of the optimal selective frame discard algorithm 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.*

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

```
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 ∉ S_i
5.      If c_j^i < c_k^i
6.        Set k = j
7.      Else If c_j^i = c_k^i and Δ_j^i > Δ_k^i
8.        Set k = j
9.  END PROCEDURE
```
(b)

```
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 ∉ S_i
5.      If Δ_j^i/c_j^i > Δ_k^i/c_k^i
6.        Set k = j
7.  END PROCEDURE
```
(c)

Fig. 6. Heuristic procedures to select the frame to discard.

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 $c_i = 1 + \frac{1}{\sqrt{d_i}}$. 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 is needed in this area to come up with a more realistic cost function based on perceptual quality of video playback [14]. 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 presented. Our algorithms can be easily modified to incorporate the specifics of other cost functions.

A. Heuristic Algorithms for JPEG Video

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 5. At each time i, the buffer is increased by $\hat{a}_i = \min(B - B_{i-1}, C)$ (line 3), as per the greedy transmission schedule. If the buffer occupancy is smaller than the size of the current frame i, i.e. $B_i + \hat{a}_i < f_i$, 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 6(a). At each time i, if the playback deadline of frame i is violated, the procedure is invoked. This 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 Δ_k^i is chosen (line 8). Finally, the selected frame k is chosen for discarding only if its gain Δ_k^i is bigger than the size of the current frame, f_i (this criterion is not shown in Figure 6(a)). Otherwise, the current frame i is discarded.

The *minimum cost* based selective frame discard algorithm, MINCD, takes the cost of discarding a frame directly into consideration. The procedure for selecting the frame to discard is given in Figure 6(b). At time i, if the playback deadline of frame i is violated, a frame k with lowest incurred cost c_k^i is chosen for discarding. Let S_{i-1} be the feasible set constructed at time $i-1$. The incurred cost c_k^i is defined to be the cost incurred if frame k is discarded at time i, i.e., $c_k^i = \phi(S_{i-1}) - \phi((S_{i-1} \cup \{i\}) \setminus \{k\})$. 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 Δ_j^i 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 Δ_j^i from discarding a frame and the cost c_j^i incurred thereof into consideration. The procedure for selecting the frame to discard is shown in Figure 6(c). It discards a frame k with the largest gain to the incurred cost ratio, i.e., Δ_j^i / c_j^i (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.

B. 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

TABLE II

CHARACTERISTICS OF JPEG VIDEO TRACES

Title	Length (min)	No. of Frames	Ave. Rate (Mbps)	Peak Rate (Unsmoothed)	Peak Rate (Smoothed)
SS	101	181457	2.28	3.99	3.30
BB	80	143442	3.04	7.29	6.54
JP	122	220061	2.73	5.73	4.78

TABLE III

COMPARISON OF SELECTIVE FRAME DISCARD ALGORITHMS FOR JPEG.

SFD Algo.	SS		BB		JP	
	Drops	Cost	Drops	Cost	Drops	Cost
JITFD	10538	15720	8778	13356	13457	20270
DISTD(2)	10272	15696	8602	13371	13141	20263
DISTD(5)	10414	15373	8692	13196	13294	19909
MINCD	10473	15332	8742	13170	13384	19845
MCMGD	10455	15246	8712	13131	13342	19747
MINFD	9907	128798	8183	106951	12516	148922

frames discarded and the cost incurred by these algorithms are compared. 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* (SS), *Beauty and the Beast* (BB) and *Jurassic Park* (JP). Table II lists the characteristics of these traces [3], 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 [13] using a client buffer size of 1 MB and zero startup delay.

Table III 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 II, 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. DISTD(2) drops 10272 frames, while DISTD(5) drops 10414 frames, larger than that of DISTD(2). However, the cost of DISTD(5) is 15373, lower than that of DISTD(2), which is 15696. 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 number of discarded frames. For the same trace, MINCD discards 10473 frames with a cost of 15332, and MCMGD incurs a cost of 15246 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. 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 7 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 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.

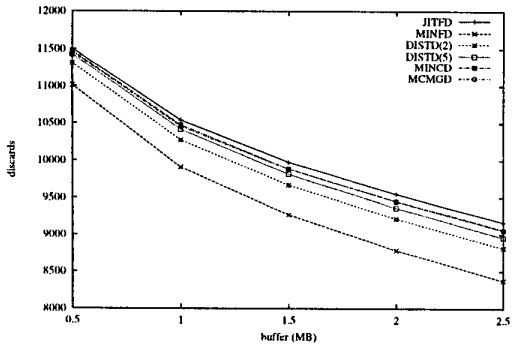
Figure 8 shows the impact of bandwidth variation for the trace *Sleepless in Seattle*. The bandwidth is varied from 2.96 Mbps to 3.12 Mbps with the client buffer size fixed at 1 MB. As the bandwidth increases, the difference in performance between the JITFD and the other four heuristic algorithms narrows 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. 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.

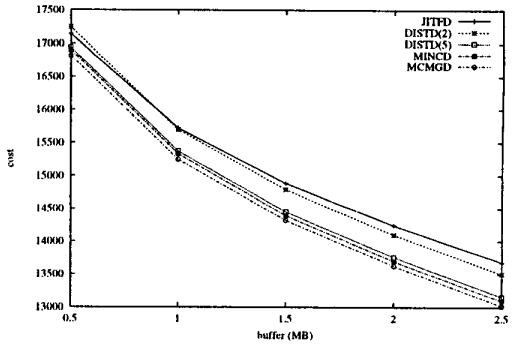
V. CONCLUSIONS

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. An optimal algorithm for solving the optimal selective frame discard problem can be designed using dynamic programming, which due to space limitation is not presented here (interested readers are referred to [16]). 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 video traces. We found that the *minimum cost maximum gain* algorithm performs best for JPEG encoded video. Extensions to these algorithms for handling MPEG videos can be found in [16].

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

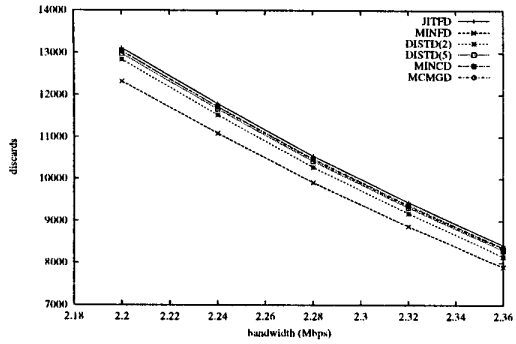


(a) buffer size *vs.* number of discarded frames

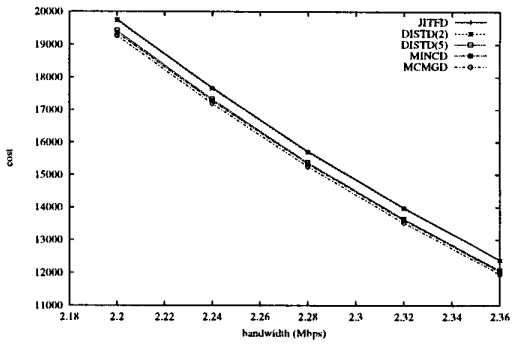


(b) buffer size *vs.* cost

Fig. 7. Performance under varying buffer sizes with C fixed at 2.28 Mbps for *Sleepless in Seattle.*



(a) bandwidth *vs.* number of discarded frames



(b) bandwidth *vs.* cost

Fig. 8. Performance under varying bandwidth with B fixed at 1MB for *Sleepless in Seattle.*

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 [1]. 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.

REFERENCES

[1] R. Aggarwal, S. Nelakuditi and Z.-L. Zhang, "Adaptive Stored Video Delivery using Selective Frame Discard across Resource Constrained Networks", Technical Report, Department of Computer Science, University of Minnesota, June 1998.

[2] W.-c. Feng, "Rate-constrained Bandwidth Smoothing for the Delivery of Stored Video", SPIE Multimedia Computing and Networking 1997.

[3] W.-c. Feng, "Video-on-Demand Services: Efficient Transportation and Decompression of Variable Bit Rate Video", Ph.D. Thesis, Univ. of Michigan, April 1996.

[4] M. Garrett and W. Willinger, "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic", Proc. ACM SIGCOMM, pp. 269-280, Aug. 1994.

[5] M. Grossglauser, S. Keshav and D. Tse, "RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic", Proc. ACM SIGCOMM, pp. 219-230, Aug 1995.

[6] C. Hsu, A. Ortega and A. Reibman, "Joint Selection of Source and Channel Rate for VBR Video Transmission under ATM Policing Constraints", IEEE Journal on Selected Areas in Communication, 1997.

[7] T.V. Lakshman, A. Ortega and A.R. Reibman, "Variable Bit-Rate (VBR) Video: Tradeoffs and Potentials," Proceedings of the IEEE, vol. 86, May 1998.

[8] J.M. McManus and K.W. Ross, "Video on Demand over ATM: Constant-rate Transmission and Transport", Proc. IEEE INFOCOM, pp. 1357-1362, March 1996.

[9] S. Ramanathan, P.V. Rangan and H.M. Vin, "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, pp. 175-187, November 1993.

[10] J. Rexford and D. Towsley, "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, November 1997.

[11] L.A. Rowe, K.D. Patel, B.C. Smith and K. Liu, "MPEG Video in Software: Representation, Transmission and Playback", IS&T/SPIE, Symp. on Elec. Imaging Sci. & Tech., San Jose, CA, February 1994.

[12] S. Sahu, Z.-L. Zhang, J. Kurose and D. and Towsley, "On Efficient Retrieval of VBR Video in a Multimedia Server", In Proc. IEEE International Conference on Multimedia Computing and Systems'97, pp. 46-53, June 1997, Ottawa, Ontario, Canada.

[13] J.D. Salehi, Z.-L. Zhang, J. F. Kurose and D. Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing", Proc. ACM SIGMETRICS, May 1996.

[14] D. Wijesekera and J. Srivastava, "Quality of Service (QoS) Metrics for Continuous Media", Multimedia Tools and Applications, Vol 2, No 3, Sept 1996, pp. 127-166.

[15] J. Zhang and J. Y. Hui, "Traffic Characteristics and Smoothness Criteria in VBR Video Traffic Smoothing", IEEE International Conference on Multimedia Computing and Systems, June 1997.

[16] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal and R.P. Tsang, "Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks", Technical Report, Department of Computer Science, University of Minnesota, July 1998.