Error Spreading: A Perception-Driven Approach to Handling Error in Continuous Media Streaming

Srivatsan Varadarajan, *Student Member, IEEE*, Hung Q. Ngo, and Jaideep Srivastava, *Senior Member, IEEE*

***Abstract*—With the growing popularity of the Internet, there is increasing interest in using it for audio and video transmission. Perceptual studies of audio and video viewing have shown that viewers find bursty losses, mostly caused by congestion, to be the most annoying disturbance, and hence these are critical issues to be addressed for continuous media streaming applications. Classical error handling techniques have mostly been geared toward ensuring that the transmission is correct, with no attention to timeliness. For isochronous traffic like audio and video, timeliness is a key criterion, and given the high degree of content redundancy, some loss of content is quite acceptable. In this paper, we introduce the concept of *error spreading*, which is a transformation technique that permutes the input sequence of packets (from a continuous stream of data) before transmission. The packets are unscrambled at the receiving end. The transformation is designed to ensure that bursty losses in the transformed domain get spread all over the sequence in the original domain, thus improving the perceptual quality of the stream. Our error spreading idea deals with both cases where the stream has or does not have inter-frame dependencies. We next describe a continuous media transmission protocol and experimentally validate its performance based on this idea. We also show that our protocol can be used complementary to other error handling protocols.**

***Index Terms*—Bursty error, error spreading, multimedia.**

I. Introduction

GROWING with the popularity of the Internet are multimedia systems and applications, as well as numerous research efforts directed at providing *continuous media* (CM) services over the Internet (e.g., video, audio). However, the Internet provides a *single class best effort* service, and does not provide any sort of guarantees [3]. A characteristic of networks of special concern to this paper is transmission errors, and specifically the dropping of data packets. Packets are dropped when the network becomes congested, and given the nature of this phenomenon, strings of successive packets are often dropped [13], leading to significant bursty errors [28]. This bursty loss behavior has been shown to arise from the drop-tail queueing discipline adopted in many Internet routers [12]. This problem may be reduced if Random Early Detection (RED) gateways are used [6]. Nevertheless, since drop-tail queueing discipline is still adopted in many routers, bursty network errors still have to be considered.

Handling bursty errors has always been problematic, especially since no good models exist for its prediction. Moreover, since most CM applications do not tolerate bursty error, it is imperative that they be handled well. Perceptual studies on CM viewing have shown that user dissatisfaction rises dramatically beyond a certain threshold of bursty error [27], [26]. These observations point quite solidly to the need for development of efficient mechanisms to control bursty errors in CM streaming.

Redundancy is the key to handling packet loss or damage in standard communication protocols. There are two main classes of error handling schemes: the *reactive* schemes and the *proactive* schemes. Reactive schemes respond by taking some action once transmission error has been detected, while proactive schemes aim at error avoidance. In reactive schemes, the reaction can be initiated by the sender or the receiver. Sender initiated reaction occurs in schemes based on feedback combined with retransmission [3], [15]. The feedback control can be based on stream rate [9], bandwidth [5], loss/delay [3], and a wide variety of network QoS parameters [14]. Receiver initiated reaction occurs in reconstruction based schemes like [25], [1]. Coding data in an error correcting manner before transmission is pro-active. Examples include the coding scheme in [7] and the *Forward Error Correction Codes* in [2]. There also exist hybridschemes which use both forms of redundancy [16]. Another technique is called *error concealment* [8], in which some form of reconstruction is done at the receiver to minimize the impact of missing data. Yet another approach is to provide real time services like RSVP and RTP, which offer varying degrees of performance guarantees for CM applications [29], [17]. Services like RTP/RSVP require that some resource allocation and/or reservation mechanism be provided by the network [3].

Recent work ([18], [30]) has proposed schemes where the overall characteristics of the data being transmitted can be used to control the transmission error. For instance, [30] has proposed selectively dropping video frames on the sender side, based on a *cost-benefit analysis* which takes into account the desired QoS. This is quite effective in a LAN (senders are known and cooperative) or the Internet if RED gateways are popular.

In this paper we propose a new type of scheme for handling bursty errors called *error spreading*. The main idea is not to reduce overall error, but rather to tradeoff bursty error (the *bad error*) for average error (the *good error*). Perceptual study of CM viewing [27], [26] has shown that a reasonable amount of overall error is acceptable, *as long as it is spread out*. A similar approach has been taken by [28]. However, the impact of packet

Manuscript received June 10, 1999; revised February 17, 2000 and November 27, 2000; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. El Zarki.

S. Varadarajan and J. Srivastava are with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: varadara@cs.umn.edu; srivasta@cs.umn.edu).

H. Q. Ngo is with Department of Computer Science and Engineering, State University of New York, Buffalo, NY 14260 USA (e-mail: hungngo@cse.buffalo.edu).

Publisher Item Identifier S 1063-6692(02)01084-1.

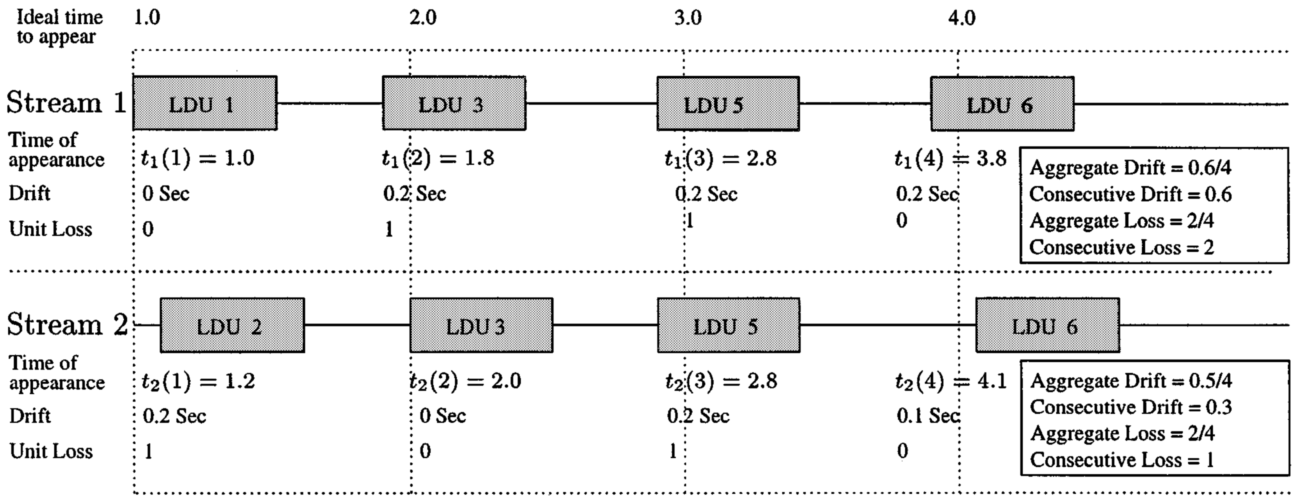

Fig. 1. Two example streams used to explain metrics.

scrambling on user perception or system resources was not investigated. Preliminary reports on the results of this paper have appeared in [11], [10], [24].

In this paper we make several contributions. First, we formulate the problem of error handling in CM transmission as a tradeoff between the user QoS requirements, network characteristics, and sender resource availability. Second, we provide a complete analytical solution for the special case where the network errors are bounded. While this solution may be of actual use only in some specialized networks, e.g., a tightly controlled real-time network, its principal use is in providing important mathematical relationships that can be used as the basis of protocols for general networks. Third, we extend the problem formulation of error handling in CM transmission as a tradeoff between the user QoS requirements, network characteristics, and sender resource availability to include streams with inter-frame dependency. Next, we use this analysis to develop such a protocol for networks where there is no bound on the error. We also provide a general error spreading technique which can handle these types of streams and a protocol which is orthogonal in nature with respect to other error handling schemes. Finally, we present results of an experimental evaluation that illustrates the benefits of the proposed scheme.

This paper is organized as follows: Section II formulates the problem and Section III presents a mathematical analysis of the bounded network error case. Section IV presents the general solution to streams with inter-frame dependencies. Section V provides a protocol where the network error is unbounded and which can be plugged into other error handling schemes. Section VI presents results of experiments and simulations which validate our claim. Finally, Section VII concludes the paper.

II. Background

This section briefly discusses the content-based continuity QoS metrics introduced in [26]. Next, we give motivations for our work. Finally, we define our problem based on the metrics introduced.

A. Perceptual QoS Metrics

For the purpose of describing QoS metrics for lossy media streams, CM stream is envisioned as a flow of data units [referred to as logical data units (LDUs) in the uniform framework of [23]]. In our case, we take a video LDU to be a frame, and an audio LDU to constitute $8000/30$, i.e., 266 samples of audio[1]. In this paper, we use mainly the content-based continuity metrics proposed in [26]. Relevant issues arising out of rates and drifts [26] are discussed briefly in Section 5.3. Note also that we shall use the term LDU and frame interchangeably.

Fig. 1 is from [26]. Given the ideal rate and the beginning time of a CM stream, there is an ideal time for a given LDU to arrive/ be displayed. Given the envisioned fluid-like nature of CM streams, the appearance time of a given LDU may deviate from this ideal. Our *drift* parameters specify aggregate and consecutive nonzero drifts from these ideals, over a given number of consecutive LDUs in a stream. For instance, first four LDUs of two example streams with their expected and actual times of appearance, are shown in Fig. 1. In the first example stream, the drifts are, respectively, 0.0, 0.2, 0.2, and 0.2 s; and accordingly it has an aggregate drift of 0.6 s per four time slots, and a nonzero consecutive drift of 0.6 s. In the second example stream, the largest consecutive nonzero drift is 0.3 s and the aggregate drift is 0.5 s per four time slots. The reason for a lower consecutive drift in stream 2 is that the unit drifts in it are more spread out than those in stream 1.

Ideal contents of a CM stream are specified by the ideal contents of each LDU. Due to loss, delivery or resource overload problems, appearance of LDUs may deviate from this ideal, and consequently lead to discontinuity. The metrics of continuity are designed to measure the average and bursty deviation from the ideal specification. A loss or repetition of a LDU is considered a unit loss in a CM stream. (A more precise definition is given in [26].) The aggregate number of such unit losses is the *aggregate loss factor* (ALF) of a CM stream, while the largest consecutive nonzero loss is its *consecutive loss factor* (CLF). In the example streams of Fig. 1, stream 1 has an aggregate loss of $2/4$ and a

[1]SunAudio has 8-bit samples at 8 kHz. An audio frame constitutes of 266 such samples, which is equivalent to a play time of one video frame, i.e., 1/30 s.

TABLE I
EXAMPLE OF HOW THE ORDER OF FRAMES SENT AFFECTS CLF

	Frame sequence			CLF
In order	01 02 03 04 05 06	07 08 09 10 11 12 13	14 15 16 17	7/17
Permuted	01 06 11 16 04 09	14 02 07 12 17 05 10	15 03 08 13	1/17
Un-permuted	01 $\boxed{02}$ 03 04 $\boxed{05}$ 06 $\boxed{07}$ 08 09 $\boxed{10}$ 11 $\boxed{12}$ 13 $\boxed{14}$ 15 16 $\boxed{17}$			1/17

consecutive loss of 2, while stream 2 has an aggregate loss of $2/4$ and a consecutive loss of 1. The reason for the lower consecutive loss in stream 2 is that its losses are more spreadout than those of stream 1. Note that the metrics already take care of losses (both consecutive and aggregate) that arise due to timing drifts.

In a user study [27], it has been determined that the tolerable value for consecutive losses of video is two frames. For audio, this limit was about three frames.

B. Impact of Network Errors on QoS

One of the most important QoS parameters that affects the quality of a CM stream is the CLF [26]. Packet transmission on networks shows a bursty behavior, alternating between spurts of packet loss and packet delivery [28], [13], [12]. This usually causes unacceptably high CLF from a user perception point of view. For example, suppose we send a sequence of 17 consecutive video frames numbered 1 to 17. During transmission, a network bursty error of size 7 occurs, which causes the loss of frames numbered 7 to 13, as shown in the first row of Table I. This causes the stream to have a CLF of $7/17$.

Now, suppose we permute this sequence of frames before transmission so that consecutive frames become far apart in the sequence, the CLF can be reduced significantly. To illustrate this idea, consider the frame transmission order shown in the second row of Table I. With exactly the same bursty error, once again seven consecutive frames are lost, except this time they are consecutive only in the permuted domain. In the original domain, these are spread far apart, as shown in row 3 of Table I. Clearly, if the 17 frames were sent in this order, we would have a CLF of only $1/17$.

Table I summarizes our example by giving three sequences and their corresponding CLFs. The first sequence is the natural order of frames, the second one is the permuted order, and the third one is the unpermuted order observed at the receiver's side. The third sequence was presented to show how the loss has been spread out over the original sequence. The boxed numbers represent lost frames.

This example raises the following question: *What permutation of the input sequence of frames minimizes the* CLF *for a given network loss?* A number of issues concerning this question need to be addressed.

- First, there is a need for a reasonable metric to compare permutations. We use the CLF metric introduced earlier, as it is directly related to human perception [26].
- Second, a number of technical questions must be answered, including:
  a) What is the minimum CLF k_0 that can be supported?
  b) How many such permutations exist?
  c) Amongst the permutations which can achieve this bound, which one do we choose?

  From an algorithmic point of view, a) is the most interesting question.
- Next, the example implicitly assumes that the frames are independent, and the loss of one frame does not affect others. For encoded streams which introduce inter-frame dependency, e.g., MPEG, merely permuting the input sequence is not sufficient.
- Finally, network behavior is nondeterministic, and any permutation scheme must address this issue.

In this paper, we address the issues listed above.

C. Problem Statement

We now formally state the problem.

1) Bursty Error Reduction Problem (BERD):

- **Objective:** to reduce the bursty error, i.e., CLF, to a perceptually acceptable level (by spreading it out over the stream).
- **Input parameters:**

— m is the sender's buffer size, in terms of LDUs. m is determined by the sender's operating environment and its current status.

— p is the upper bound on the size of a bursty loss in the communication channel, within a window of m LDUs.

— k is the user's maximum acceptable CLF.

- **Output:** a permutation function f on S, where $S = \{1, 2, \ldots, m\}$, which decides the order in which a set of m consecutive LDUs must be sent. Moreover, the system is expected to give the lower bound k_0 which is the minimum CLF that can be supported in this constrained environment.
- **Assumption:** two consecutive bursty losses are at least m LDUs apart.

Fig. 2 visualizes how the solution space for a particular value of p would appear. The boundary of the curve is essentially what we found. Above it is the feasible region, where intuitively if we increase m, then k should still be the same or less. There is a typical trade off between buffer size m and CLF k. The greater m is, the less k we can support but also the greater memory requirement and initial delay time. Given m_0, line $m = m_0$ cuts the boundary curve at k_0, at or above which we can support. Conversely, given k_0, line $k = k_0$ intersects the curve at m_0, the minimum buffer size required to support k_0.

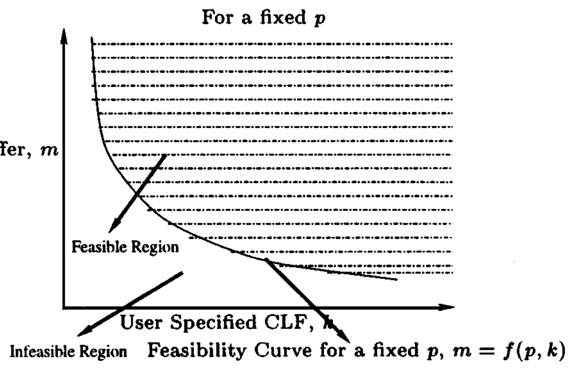

Fig. 2. Part of deterministic solution space.

There are several points worth noticing. First, we deal only with data streams that have no inter-frame dependency such as Motion-JPEG or uncompressed data streams (audio, video, sensor data, ...). This allows us to consider every frame to be equally important; thus, we can permute the frames in any way we would like to. Second, the frames in these types of streams have relatively comparable sizes. For example, a sequence of MJPEG frames only has a change in size significantly when the scene switches. So, no matter if we send the frames by breaking them up into equal size UDP packets or if it is the transport layer interface (TLI) which does so, a consecutive packet loss implies a proportional consecutive frame loss. Finally, to satisfy our assumption that two consecutive lost windows are at least m frames apart, closer lost windows can be combined and consider to be a larger lost window.

III. Error Spreading Through Permutations

In this section, we will discuss the case where the bound p of continuous network loss is known. For convenience, we first state the problem in purely mathematical terms and establish some notations to be used throughout the proof.

We are given positive integers m and p. Let S_m denotes the set of all permutations on $[m] = \{1, 2, \ldots m\}$. For any permutation $\pi \in S_m$, the sets W_i^π given by

$$W_i^\pi = \{\pi_i, \pi_{i+1}, \ldots \pi_{i+p-1}\}, \quad 1 \le i \le m$$

are called the *sliding windows* of size p (of π), where the indices are calculated modulo m, then plus 1. The reader is referred to the Appendix for detailed examples. Thus, when $1 \le i \le m-p+1, W_i^\pi = \{\pi_i, \pi_{i+1}, \ldots \pi_{i+p-1}\}$, and when $m-p+1 < i \le m, W_i^\pi = \{\pi_i, \ldots, \pi_m, \pi_1, \ldots, \pi_{i+p-m-1}\}$.

For any pair of integers k and l such that $1 \le k < l \le m$, let $[k, l]$ denotes the set $\{k, k+1, \ldots l\}$. Let $\{c_i^\pi\}_1^m$ be the sequence of integers defined as follows.

$$c_i^\pi = \begin{cases} \max\{|[k,l]|, [k,l] \subseteq W_i^\pi\} \\ \qquad \text{if } 1 \le i \le m-p+1 \\ \max\{|[k,m]| + |[1,l]|, \\ \quad [k,m] \subseteq \{\pi_i, \ldots \pi_m\}, \\ \quad [1,l] \subseteq \{\pi_1, \ldots \pi_{p+i-m-1}\}\} \\ \qquad \text{if } m-p+1 < i \le m \end{cases}$$

Let $C^\pi = \max\{c_i^\pi \mid 1 \le i \le m\}$. Then k_0 is defined to be

$$k_0 = \min\{C^\pi, \pi \in S_m\}$$

Our objective is to find k_0 as a function of m and p. Moreover, we also wish to specify a permutation π so that $C^\pi = k_0$.

Informally, when $1 \le i \le m-p+1, c_i^\pi$ is the maximum number of consecutive integers in W_i^π. While if $m-p+1 < i < m, c_i^\pi$ is the sum of two quantities x and y, where x is the length of the longest consecutive integer sequence in $\{\pi_i, \ldots, \pi_m\}$ which ends in m, and y is the length of the longest consecutive integer sequence in $\{\pi_1, \ldots, \pi_{i+p-m-1}\}$ which starts at 1. The reason for this is that suppose we apply our permutation to two adjacent buffers of size m, we would like our permutation to also deal with the case where the network loss burst occurs across these two buffers.

The value of k_0 and permutation π depends tightly on the relationship between p and m. We summarize our result in Theorem 3.2. The details of the proof are given in the Appendix. Algorithm *calculatePermutation* (m, p) produces the appropriate permutation which supports the best k_0 given m and p.

Remark 3.1: If p is known and m is fixed, then $k_0 = 0$ when $p = 0$ and $k_0 = m$ when $p \ge m$.

Theorem 3.2: If p and m are both determined, then

- $k_0 = 0$ when $p = 0$ and $k_0 = m$ when $p \ge m$
- $k_0 = \lfloor (p/(m-p+1)) \rfloor + 1$ when $0 < p < m$.

Proof: Since if $0 < p \le (m/2)$ then $\lfloor (p/(m-p+1)) \rfloor = 0$, this follows from the Lemmas given in the Appendix. □

Also note that if the desired k_0 is given, these formulas allow us to find the minimum buffer size m_0 to achieve k_0.

Algorithm *calculatePermutation* (m, p) is a permutation generator (listed in Fig. 3) which generates permutation π with $C^\pi = k_0$ on input m and p. Notice that it takes only linear time.

1) Benefits of Solving the Bounded Error Case: The assumption that p is known can be envisioned in future networks where some sort of QoS guarantees are provided, such as ATM, Internet2, etc. More importantly, it gives us a rigid background to solve the unbounded error case.

IV. Modeling Streams With Inter-Frame Dependency

This section is organized as follows. Section IV-A defines related concepts from Combinatorics, Section IV-B discusses our solution for MPEG streams, and Section IV-C generalizes the analysis for any stream with inter-frame dependencies.

A. Poset Model and Its Properties

The inter-frame dependency of a stream can be modeled by a combinatorial structure called a partially ordered set (*poset*). For a rigorous treatment on posets, the reader is referred to [22].

A poset is a set P along with a binary relation $\rightarrow$ satisfying the following axioms:

- *Reflexivity:* $\forall x \in P, x \rightarrow x$.
- *Antisymmetry:* if $x \rightarrow y$ and $y \rightarrow x$ then $x = y$.
- *Transitivity:* if $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$.

```
calculatePermutation(m, p)
  if $p \leq 0$ or $p \geq m$ then
    output the identity permutation
  end if
  if $p \leq \frac{m}{2}$ then
    $M \leftarrow \{j,\ p \leq j \leq \frac{m}{2} \wedge \gcd(m,j) = 1\}$
    if $M \neq \emptyset$ then
      $p' \leftarrow \min\{j,\ j \in M\}$
      for $i \leftarrow 1$ to $m$ do
        $\pi(i) \leftarrow ((i-1)p' \bmod m) + 1$
      end for
    else
      ** $M = \emptyset$, $m$ must be even **
      $p' = \frac{m}{2}$
      for $i \leftarrow 1$ to $m$ do
        $\pi(i) \leftarrow p'.(i \bmod 2) + \lceil \frac{i}{2} \rceil$
      end for
    end if
  else
    $q \leftarrow m - p$
    $r \leftarrow \left\lfloor \frac{p}{q+1} \right\rfloor$
    $t \leftarrow \left\lfloor \frac{m}{r+2} \right\rfloor$
    $t' \leftarrow m \bmod (r+2)$
    if $t' = r + 1$ then
      for $i \leftarrow 1$ to $t + 1$ do
        $a_i \leftarrow 1 + (i-1).(r+2)$
        $b_i \leftarrow (r+1) + (i-1).(r+2)$
      end for
      $C \leftarrow \{1, 2, \ldots m\} - \{a_i\} - \{b_i\}$
      $C$ is extracted in increasing order.
      for $i \leftarrow 1$ to $t + 1$ do
        $\pi(a_{t+2-i}) \leftarrow i$
      end for
      for $i \leftarrow t + 2$ to $m - (t+1)$ do
        $\pi(c_{m-i-t}) \leftarrow i$
      end for
      for $i \leftarrow m - t$ to $m$ do
        $\pi(b_{m-i+1}) \leftarrow i$
      end for
    else
      ** i.e. $0 \leq t' \leq r$ **
      for $i \leftarrow 1$ to $t$ do
        $a_i \leftarrow t' + 1 + (i-1).(r+2)$
        $b_i \leftarrow i.(r+2)$
      end for
      $C \leftarrow \{1, 2, \ldots m\} - \{a_i\} - \{b_i\}$
      $C$ is extracted in increasing order.
      for $i \leftarrow 1$ to $t$ do
        $\pi(a_{t+1-i}) \leftarrow i$
      end for
      for $i \leftarrow t + 1$ to $m - t$ do
        $\pi(c_{m-i-t+1}) \leftarrow i$
      end for
      for $i \leftarrow m - t + 1$ to $m$ do
        $\pi(b_{m-i+1}) \leftarrow i$
      end for
    end if
  end if
```

Fig. 3. Permutation generator algorithm.

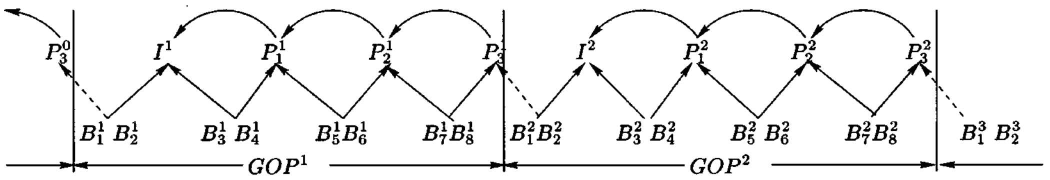

Fig. 4. Sample pattern of MPEG frame dependency.

Abusing notation, we shall also call the poset P. Notice that a poset can also be represented by a *directed acyclic graph* (DAG). Two element x and y of a poset P are *comparable* if $x \rightarrow y$ or $y \rightarrow x$. Otherwise, x and y are *incomparable*. A *chain* is a poset in which any two elements are comparable. A chain has length n if it has $n+1$ elements. An *antichain* is a subset A of a poset P such that every pair of elements in A is incomparable. An *antichain decomposition* of size w of P is a partition of P i nto disjoint antichains $A_1, \ldots, A_w$. For $x, y \in P, y$ covers x if $x \rightarrow y$ and for all $z \in P, x \rightarrow z$ and $z \rightarrow y$ implies either $z = x$ or $z = y$. $x \in P$ is *minimal* if $\not\exists z \in P, z \neq x, z \rightarrow x$. P is *ranked* if there exists a unique rank function $\rho : P \rightarrow \{0, 1, 2, \ldots n\}$, where n is the length of the longest chain in P, such that if $x \in P$ is minimal then $\rho(x) = 0$, and if y covers x then $\rho(y) = \rho(x) + 1$. A *linear extension* of a poset P is an *order preserving bijection* ϕ which maps P onto a chain of size $|P|$. Note that this is similar to a *topological sort* of a DAG [4]. Thus, a linear extension of a poset can be obtained by a topological sort algorithm.

B. Analysis of the MPEG Case

Fig. 4 is a typical dependency diagram amongst MPEG video frames. A group of pictures (GOP) is a set of consecutive frames beginning with an I-frame (inclusive) and ending with the next I-frame (exclusive). Although not required by the MPEG standard, a fixed spacing between I-frames and between *anchor frames* (I- and P-frames) is often used. Thus, usually all GOPs have the same size. In Fig. 4, the GOPs have size 12. The broken arrows represent possible dependency of the beginning B-frames of the current GOP to the last P-frame of the previous GOP. This is called *open* GOP. MPEG allows *closed* GOPs where there is no such dependency. Observe that if the delivery of *anchor pictures* (I and P frames) can be ensured, then the *nonanchor pictures* (B frames) are free to be permuted. In real-time video transmission, timeliness of frame delivery is important, and thus, permuting frames before sending helps in reducing CLF, as well as intelligent frame dropping when transmission is lagging behind in time.

CM streams with inter-frame dependency can be modeled as a poset P of frames, where for any two frames x and $y, x \rightarrow y$ if and only if x is dependent on y (directly or indirectly). Borrowing terminology from MPEG, a frame $y \in P$ is called an *anchor frame* if $\exists x \in P, x \rightarrow y$. In multimedia transmission, due to limited buffer capability of the sender, at any time only a subposet of P is present in the buffer. Moreover, it is reasonable that the frame transmission order be a linear extension of P where the anchor frames go first, since the nonanchor frames cannot be reconstructed without the anchor frames.

To illustrate this, consider Fig. 4 where the anchor frames are the I's and P's. Assuming the server buffer has capability for two GOPs, a possible order of frame transmission is

$$\ldots I^1, B_1^1, B_2^1, P_1^1, B_3^1, B_4^1, P_2^1, B_5^1, B_6^1, P_3^1, B_7^1, B_8^1,$$
$$I^2, B_1^2, B_2^2, P_1^2, B_3^2, B_4^2, P_2^2, B_5^2, B_6^2, P_3^2, B_7^2, B_8^2, \ldots$$

If all frames are received in this order, the client buffer has high utilization. At any time, the client's decoder needs to store only two anchor frames and the scratch memory for the frame being currently decoded. For example, suppose P_3^0 and I^1 are in the decoder memory. Upon arrival B_1^1 is loaded into the scratch memory for decoding, followed by B_2^1, etc; When P_1^1 arrives, it is decoded and replaces P_3^0; and the process continues.

In practice, however, network losses can occur, frames can arrive out of order, and the frames in the server's buffer can only be sent within a buffer cycle time. These can potentially lead to very high CLF, and thus unacceptable perceptual quality. For example, consider the case where P_3^2 is lost, either due to network

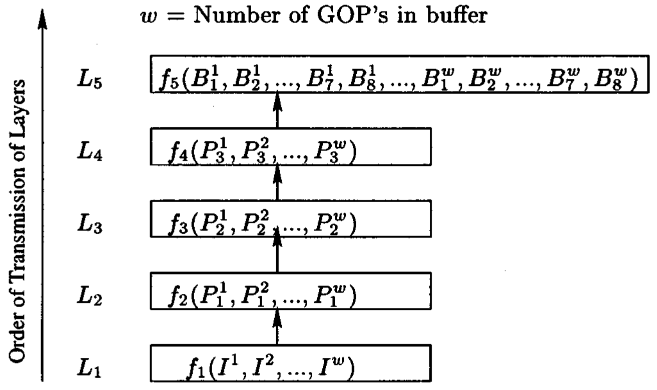
w = Number of GOP's in buffer
Order of Transmission of Layers
L5
f5(B^1_1, B^1_2, ..., B^1_7, B^1_8, ..., B^w_1, B^w_2, ..., B^w_7, B^w_8)
L4
f4(P^1_3, P^2_3, ..., P^w_3)
L3
f3(P^1_2, P^2_2, ..., P^w_2)
L2
f2(P^1_1, P^2_1, ..., P^w_1)
L1
f1(I^1, I^2, ..., I^w)

Fig. 5. Layered permutation transmission order for MPEG.

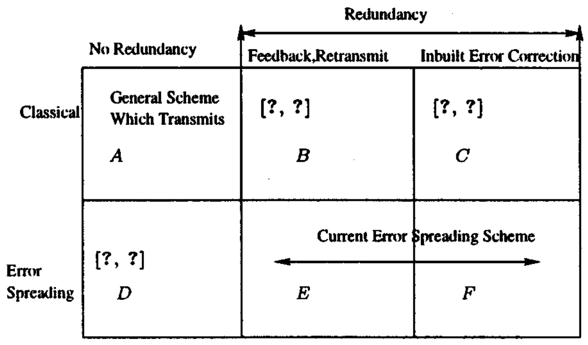

Fig. 6. Classification of error handling schemes.

loss or the server running out of buffer cycle time. B^2_7, B^2_8, B^3_1, and B^3_2 are all considered to be lost since we need P^2_3 to reconstruct them. This causes a CLF of 5—more than twice the perceptually tolerable level of CLF [27]. The reason for this is clear, namely if any anchor frame is lost, a high CLF automatically occurs.

To solve this problem, we transmit the anchor frames first: $I^1, I^2, P^1_1, P^2_1, P^1_2, P^2_2, P^1_3, P^2_3$, and also try to ensure their arrivals, possibly with retransmission. After all the anchor frames have been sent, the B frames are permuted using the error spreading scheme and transmitted in that order. Thus, if there isn't enough time to send all the B frames, this ordering allows the selection of which frames to send. Note that the anchor frames usually have larger sizes and thus take more time for transmission. However, given the nature of dependencies, their successful transmission is critical.

Further, suppose the sender and receiver can handle buffering of w GOPs, for some natural number w. Now, it is possible to permute the I-frames in the buffer, permute the P^i_1's $1 \leq i \leq w$, permute P^i_2's $1 \leq i \leq w$, etc. The advantage is that if all slots are used up before all anchor frames have been successfully transmitted, the CLF is better than without permutation. This scheme is the *Layered Permutation Transmission Order for MPEG* and is illustrated in Fig. 5. Here, f is the permutation function generated by the *calculatePermutation* algorithm.

Notice that the dependencies we are considering here are at the frame level. Analyzing the effect of our scheme with respect to finer inter-object dependencies, such as spatial and/or temporal dependencies in MPEG-4,[2] is beyond the scope of this paper. However, as MPEG-4 provides a facility for combining individual visual objects into a scene using a hierarchical description, it is conceivable that the generalized solution presented in Section IV-C is applicable.

C. Generalized Solution

In general, given a CM stream whose inter-frame dependency is modeled by a poset P, a general solution must answer the following questions: 1) which sets of frames can be permuted before transmission, and 2) which sets of frames form each layer in the proposed scheme. To answer 1), notice that the transmission order of frames must obey the dependencies in P. Thus, the permutable sets are precisely the antichains of P. To answer 2), observe that **if** there exists an antichain decomposition $A_1, A_2, \ldots A_w$ of P such that $1 \leq i < j \leq w, \Rightarrow \forall x \in A_i, \forall y \in A_j,\ y \not\to x$ **then** we can use these antichains to be the layers of the transmission scheme. Intuitively, frames in A_j have higher priority than those in A_i since the latter might depend on the former, but not vice versa. A natural goal is to minimize the number of layers, increasing the average number of frames in each layer, which can make the permutation more effective. If P is ranked, as is the case with MPEG and H261, the situation becomes particularly easy. Let A_i be the set of elements of P with ranks $i, 0 \leq i \leq w$, where w is the length of the longest chain in P. Then, clearly for all i, A_i is an antichain. Moreover, it is a well known fact from poset theory that the size of the minimum antichain decomposition is equal to the size of the longest chain [22]. Consequently, P being ranked automatically gives us the best antichain decomposition from which the layering can be derived.

V. Error Spreading Protocol

Perceptual studies have established the tolerance levels for discontinuity in CM viewing [27], [26]. Discontinuities beyond these levels has been shown to increase user dissatisfaction to unacceptable levels. The proposed scheme uses an end-user driven approach to minimize the amount of discontinuity perceived. Error handling schemes can be classified into the different categories shown in Fig. 6. Our earlier work [10], [11] has shown that the scrambling scheme decreases CLF, the measure of a CM stream's discontinuity, when network loss is bursty. This proves that the scheme proposed in block D is better than the naive transmission in block A of Fig. 6. As can be seen from Section III, schemes in block A or D work for encoding schemes like MJPEG, with no inter-frame dependency. However, for a dependent stream like MPEG, where loss of a critical frame (e.g., MPEG I frames) cannot be tolerated, the schemes in blocks A and D are not very useful.

A variety of schemes have been proposed in the literature (block B and C in Fig. 6) which use some form of redundancy to handle transmission failure in streams with inter-frame dependency. Reactive schemes use feedback/retransmission based (block B), while proactive schemes use forward error correction in the transmitted stream (block C). Additionally, there are hybrid schemes which combine both proactive and reactive approaches [16]. In Section IV, we provided a permutation order

[2]See ISO/IEC 14 496-1, Coding of audio-visual objects: systems, final draft international standard, ISO/IEC JTC1/SC29/WG11 N2501, October 1998.

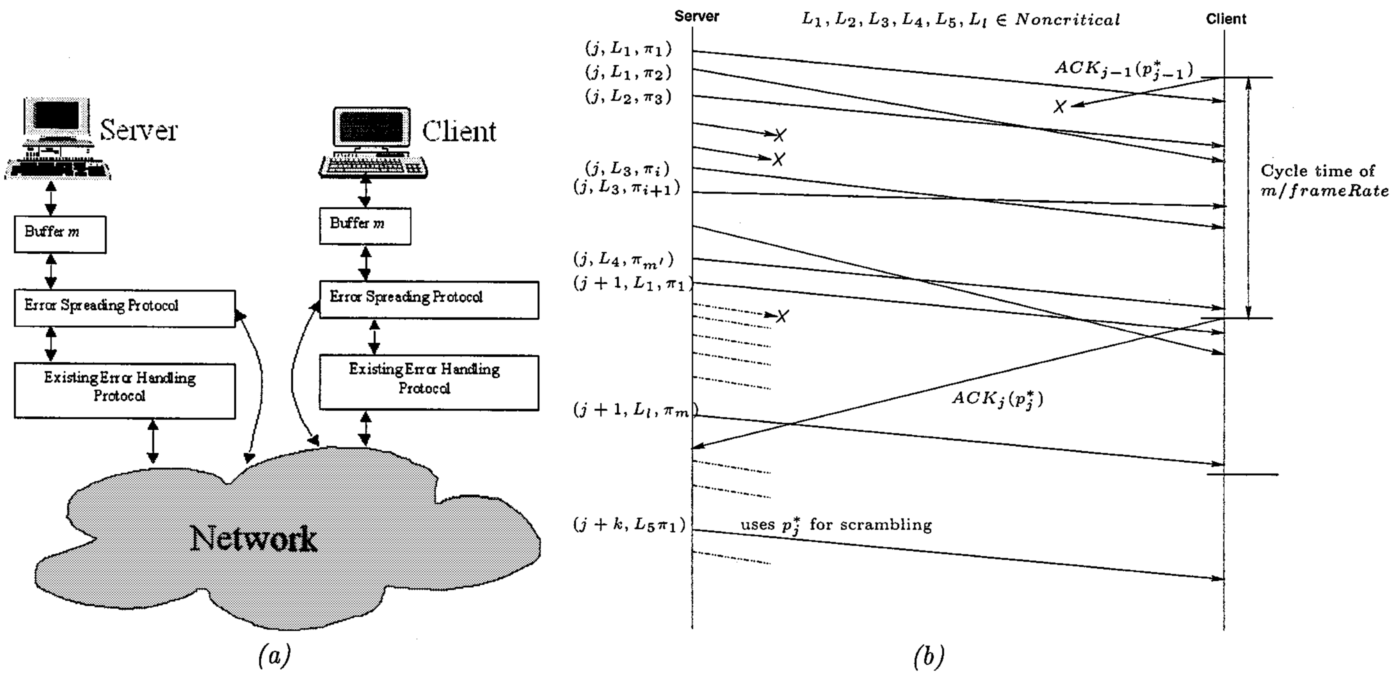

Fig. 7. (a) Protocol setting. (b) Protocol illustration.

for dependent streams, e.g., MPEG, which improves CLF. The goal of this section is to show that error spreading techniques can be used orthogonally to the error handling schemes in blocks B and C, which will help reducing CLF as much as possible.

This section is organized as follows. Section V-A addresses the buffer requirement for the use of our protocol. Section V-B discusses how our protocol functions. Section V-C argues that our protocol is practical, in terms of delay introduced, for all noninteractive applications. Lastly, Section V-D establishes the complementary nature of our protocol with other error handling schemes. The rest of this section uses MPEG as an illustration, though our technique can be applied to any dependent stream.

A. Buffer Requirements

As shown in Fig. 7(a), the server and client both require a buffer of size $m = w * F_{\text{GOP}}$, where $F_{\text{GOP}} =$ maximum length of a GOP, and $w =$ number of GOPs, $w \geq 2$. Table II gives GOP statistics for different MPEG traces (traces can be found at ftp://ftp.cs.umn.edu/users/varadara/Traces. The data has traces with GOP size 15 (30 frames/s) as well as GOP size 12 (24 frames/s). Observe that the largest GOP size is 932 710 bits or 113 kbytes. Thus, $m = 2 * F_{\text{GOP}} = 2 * 113 = 226$ kbytes is quite viable.

B. Error Spreading Protocol

In the proposed layered permutation transmission order, L_i denotes the set of frames in layer $i, 1 \leq i \leq l$, where l is the number of layers. As discussed earlier, l is also the size of the antichain decomposition. In the example of Fig. 5, $l = 5$. A layer is *critical* if it contains anchor frames, on which other frames depend. Let the set of Critical Layers be Critical $= \{L_1, L_2, \ldots, L_q\}$ where $1 \leq q \leq l$. Hence, Noncritical $= L -$ Critical $= \{L_{q+1}, \ldots, L_l\}$. In Fig. 5, Critical $= \{L_1, L_2, L_3, L_4\}$ and Noncritical $= \{L_5\}$. Since the

TABLE II
GOP DATA INFORMATION FOR MPEG TRACES

GOP Size Info	Avg (bits)	Devn (bits)	Max (bits)	Min (bits)	# GOP
Jurassic Park	156931	62982.7	627776	49664	3333
Silence of the Lambs	87731.3	52986.6	462056	26632	3333
Starwars	187211	72471.2	932710	77754	14504
Terminator	130851	45168	407512	24840	3333
Beauty and the Beast	600552	20103.4	769376	462336	10791

successful receipt of critical layers is important, the transmission order of layers is $L_1, L_2, \ldots, L_l$. In the proposed scheme, frames in L_i are scrambled using the function $f_i()$ which is generated by the call *calculatePermutation* $(|L_i|, p)$, whose algorithmic details are given in Section III. $|L_i|$ is the number of frames in L_i and p is the upper bound on network bursty loss within a window of $|L_i|$ LDUs.

Some CM systems use TCP/IP for communication. However it has been shown that CM applications based on TCP are unstable when the real time bandwidth requirements are larger than the available bandwidth [21]. Thus, the proposed protocol uses the UDP communication model, as in [19], [20], with feedback for loss estimation. We assume that m, the buffer size, and the GOP pattern is known in advance by both client and server. This can be obtained by an initial negotiation.

At the server side, a buffer of size m is kept. Server permutes frames (actually frame indices) based on current set of parameters p_i, which is the upper bound on the bursty network loss within L_i, and then initiates transmission of the frames in the buffer. Server changes the permutation scheme based on client's feedback for only the noncritical layers, and uses a fixed permutation for critical layers. Frames in layers $L_i \in$ Critical are retransmitted upon a loss. Alternatively, a forward error correction mechanism may be used. So a feedback on the loss rate p_i

(in a window of $L_i \in \text{Critical}$) for these frames can be avoided. Frames of a critical layer are permuted using a function generated by *calculatePermutation* for $k_0 = 1$ and $0 < p \leq m/2$. This ensures lower CLF for the *Critical* layers. In the case of noncritical layers, the permutation scheme is changed only at the start of the next buffer of m frames. Thus, we ensure *minimal feedback* needed to *reduce CLF*.

It takes $\text{CycleTime} = m/\text{frameRate}$ to fill up the client's buffer (frame rate is typically 30 or 24 frames/s). The client knows the GOP pattern of the transmitted frames (because of initial negotiation). It keeps track of the previous window's estimate of loss rate for all layers $L_i \in \text{Noncritical}$. It waits for *Cycle Time* and transmits the next estimated loss rate p_i for all layers $L_i \in \text{Noncritical}$ to the server. It sends feedback (ACK) in a UDP packet. Note that ACK packet is also given a sequence number so that out of order ACK packets will be ignored. The server makes its decision based on the maximum sequence numbered ACK.

Denote $p_{n,i}$ as the actual bursty network loss and $p^*_{n,i}$ as the estimated loss rate, in $L_i \in \text{Noncritical}$ for the nth window of m LDUs. Initially, when $n = 1$, the server assumes the average case where $p_{n,i} = \lfloor (|L_i|)/(2) \rfloor$. We use exponential averaging to estimate the next loss. Suppose we are currently at the nth window, $p^*_{n,i}$ is determined by

$$p^*_{n,i} = \lceil \alpha . p_{n,i} + (1 - \alpha) . p^*_{n-1,i} \rceil. \quad (1)$$

We have picked $\alpha = (1/2)$, since we consider the current network loss and the average past network loss to be equally important. This value works reasonably well, as shown in Section VI.

Fig. 7(b) illustrates an example of how client and server interact during transmission of layers $L_i \in \text{Noncritical}$. The transmission of layers $L_i \in \text{Critical}$ is easily seen as the server *does not require client feedback information* (when our protocol is used alone) and hence is not illustrated. In Fig. 7(b), there is no indication of redundancy (retransmissions/error correcting code). (j, L_3, π_i) is the time when the sender sends the ith permuted frame of layer L_3 on jth buffer window. Note that it might not be possible to transmit all m frames (thus, all l layers) because some frame slots have been used for retransmission of lost frames from critical layers. The client sends back ACK_j containing the estimated p^*_j for all layers in buffer window j. By the time server gets ACK_j, it could be in the $(j + k)$th buffer window. So, it uses ACK_j for the $(j + k)$th buffer window. Finally, if ACK_{j-1} is lost, its feedback information has not been used for transmission of any subsequent buffer windows. **Note:** For streams which have no dependency (like MJPEG), the above protocol simplifies to just a scrambling of frames and estimating loss rate p for the whole window m, details of which are given in [10]. Also, as shown earlier our approach works for any encoding scheme, even though the illustrations are with MPEG.

C. Delay Factors

Delay factors typically include startup delay and individual timing drifts (as defined in Section II-A). There are good reasons to be concerned about the delay and/or drift incurred by our scheme, especially for real time applications such as VoD or Internet telephony. However, in this subsection we argue that the scheme does not introduce new drifts and the only delay it induces is the startup delay, which is small enough for most practical purposes, except in the case of highly interactive applications. In Section VI-B, we shall also demonstrate our point by experimentally showing that no new drifts are introduced.

As described above, the sender and receiver's buffers are used as a method to smooth out bursty network losses. In effect, we only shift back the media playing time by a constant amount of time (startup delay), i.e., *delay is not cumulative*. Frames that do not arrive within the buffer cycle time are considered to be lost, which is typical in all real-time multimedia applications. Our scheme clearly does not introduce new drift factors. It only requires a startup delay time d which could be estimated as

$$d = \text{time to fill up buffer} + \text{time to send frames in buffer} + \frac{\text{rtt}}{2}.$$

Notice that d is not the *additional* amount of delay introduced by our scheme. Any real-time transmission protocol would have more or less the same delay formula. The additional delay comes from the extra buffer size. However, the overall startup delay is still very much viable, even when there are inter-frame dependencies. As an example, let us revisit Table II. Suppose we use a buffer of two GOPs with 15 frames per GOP. Further assume that our bandwidth is 10 Mb/s and $\text{rtt} = 23$ ms, then the startup delay we need even for the largest GOP (Star Wars, 932 710 bits) is

$$\begin{aligned} d &= 2 \times 1/2 + 2 \times 932\,710/10^7 + 23/(2 \times 1000) \\ &\approx 1.2 \text{ s.} \end{aligned}$$

The number for four GOPs is roughly doubled, i.e., 2.4 s, which is clearly viable for most purposes. For audio data, the startup delay is clearly a lot less and thus it is not of concern here.

Notice that we do not use the buffer for prefetching and/or smoothing rates, etc. These mechanisms can be used in conjunction with our scheme, the manner of which is discussed in the next section. It is conceivable that we need finer control of the startup delay time. This would require bandwidth/delay estimation and could be part of the negotiation phase in our scheme, whose discussion is omitted in this paper.

D. Error Spreading as an Orthogonal Dimension

As seen from the earlier sections, we have parameterized the different characteristics of the *error spreading protocol*. Error spreading itself does not require any bandwidth/delay estimation and uses only loss rate estimation as shown above. Hence, it is possible to build an *error spreading module* which uses a control channel for loss rate estimation, independent of any other error handling protocol [refer Fig. 7(a)]. It is clear that any error handling protocol using retransmission or forward error correcting codes, e.g., [3], [15], [25], [2], [16], or error concealment protocol [8], only decreases the *Critical* set and increases the *Noncritical* set. Thus, knowing the frame reconstruction mechanism available on the client side and/or the retransmission protocol only fixes the number q of critical layers. To incorporate the

TABLE III
8 FRAMES ORDERING OF IBO AND ONE OF THE CASES OF OUR SCRAMBLED ORDER

	Frame sequence		Consecutive Loss/ Window Size
In order	01 02 03	04 05 06 07 08	5/8
IBO	01 05 03	07 02 06 04 08	3/8
Our order	01 04 07	02 05 08 03 06	2/8

error spreading technique into other error handling protocols, the latter schemes have to implement the following features:

- Error handling protocol now works with w GOPs instead of 1, i.e., the start up delay increases to w/N_{GOP} where N_{GOP} is the number of GOPs displayed in 1 s.
- The sender needs to pick up the frames to be transmitted from the error spreading module's buffer.
- The receiver informs the client side error spreading module of the received frames in a buffer of m LDUs.
- The transmission decision and schedule for a frame in the error handling protocol (i.e., for frames I, B, P in MPEG) is now done for $|L_i|$ frames in each layer L_i.

To investigate the feasibility of implementing the above features, the error spreading scheme was implemented on the *Berkeley Continuous Media Toolkit* (CMT) [20]. A description on CMT and the changes that was made to it to incorporate our scheme is given in [11]. Frame priority in CMT is calculated in a manner similar to our layered scheme. All I frames have highest priority, P frames are lower, and B frames are lowest. I frames and P frames might have to be retransmitted if they are lost and time still permits. The set of B frames are prioritized based on the Inverse Binary Order or IBO (which was quoted in CMT code as by Daishi Harada), example in Table III. Note that frame dropping can potentially occur at any of the previously mentioned objects, if the object decided it cannot transmit the given frame in the estimated available resources or if a frame playback deadline has elapsed.

We changed IBO to our error spreading algorithm presented earlier. Also, CMT provides a handle onto the buffer size, parameter W, by allowing the user to vary the cycle time of a bunch of frames. Losses of B frames occur only in the tail of the set of B frames because of the way CMT's protocol works. It sends a bunch of frames at the head of the buffer (which has I, P and B frames in that order), the number of which depends on its estimated parameters. As can be seen in Table III, in a pathological network scenario wherein the number of frames lost is greater than half the number of B frames sent, IBO performance starts decreasing, while in our scheme we still provide better CLF. Further, in our layered orders, we use our ordering mechanism among the I and P frames ensuring more number of consecutive GOPs are received at the client in a highly lossy network condition. As shown earlier, our approach is not restricted to any encoding (not just MPEG). We have provided the details in Section IV. Thus our model and approach can be extended to any type of continuous media and its delivery, reducing CLF even in a pathological network scenario.

TABLE IV
SUMMARY DATA FOR THE VIDEO CLIPS

	Frame size range	Median	Mean	Std devn	# frames
Clip 1	5276-36364	9544	10845	4450.7	2607
Clip 2	5072-34408	10282	10916	3642.8	1736

VI. EXPERIMENTAL EVALUATION

The following two sections present the evaluation of our scheme in two scenarios. In Section VI-A, we present two cases of experiments done over streams with no inter-frame dependency. In one case the protocol has been implemented and tested over a long haul network. In the second case, we use a data set extracted from a real-time application such as *Internet Phone* and simulate our protocol. We show the reduction in CLF in both the cases. Our protocol has smoothed out CLF to be within the range of perceptually acceptable tolerance. Also, it adapts quite well with abnormality in network loss pattern. Moreover, *almost all* CLF values are within the range of perceptual tolerance (see Section II-A). In Section VI-C, we present the results of a detailed simulation-based evaluation of our protocol using the layered permutation transmission order for MPEG described in Section IV-B, and compare it against the usual MPEG transmission model.

A. Experiments

1) Video Experiment: Media Delivery Over a Long Haul Network: We have conducted experiments of sending two MJPEG video clips over LAN and WAN. Due to limited space, only the result of WAN is shown here. However, the behavior of our protocol is the same in both cases. We transfered data from rawana.cs.umn.edu, a UltraSparc in the Computer Science department, University of Minnesota to lombok.cs.uwm.edu, a SunSparc in the Computer Science department, University of Wisconsin, Milwaukee. The experiment was conducted at 9:45 am when network traffic is expected to be average. Both clips have resolution 512 × 384. Other information about the clips is summarized in Table IV.

Our buffer window is of size 50. Three times "traceroute" told us that the packets typically go through 14 hops in between. A sample traceroute session is as follows.

```
1 eecscix.router.umn.edu (160.94.148.264) 2 ms 1 ms 1 ms
2 tc8x.router.umn.edu (128.101.192.254) 23 ms 4 ms 3 ms
3 tc0x.router.umn.edu (128.101.120.254) 6 ms 1 ms 1 ms
4 t3-gw.mixnet.net (198.174.96.5) 1 ms 1 ms 1 ms
5 border5-hssi1-0.Chicago.cw.net (204.70.186.5) 11 ms 11 ms 29 ms
6 core2-fddi-0.Chicago.cw.net (204.70.185.49) 11 ms 11 ms 11 ms
7 core2-hssi-3.WillowSprings.cw.net (204.70.1.225) 13 ms 13 ms 15 ms
8 core3.WillowSprings.cw.net (204.70.4.25) 310 ms 52 ms 123 ms
9 * ameritech-nap.WillowSprings.cw.net (204.70.1.198) 245 ms 35 ms
10 aads.nap.net (198.32.130.39) 18 ms 18 ms 21 ms
11 r-milwaukee-hub-a9-0-21.wiscnet.net (207.112.247.5) 25 ms 22 ms 27 ms
12 205.213.126.39 (205.213.126.39) 19 ms 20 ms 23 ms
13 miller.cs.uwm.edu (129.89.139.22) 24 ms 25 ms 21 ms
14 lombok.cs.uwm.edu (129.89.142.52) 24 ms * 25 ms
```

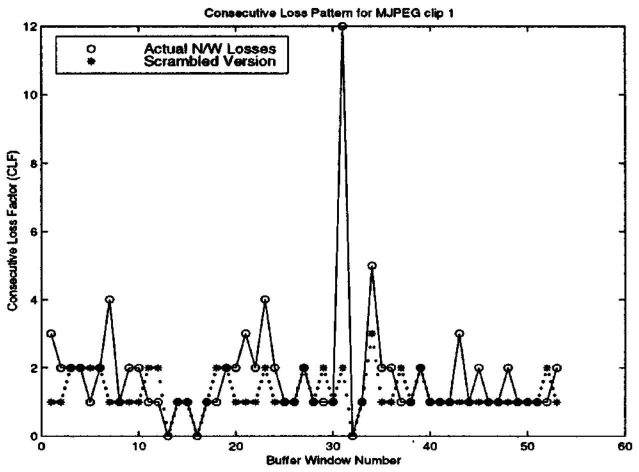

Fig. 8. Performance of our protocol when transmitting video over long-haul network.

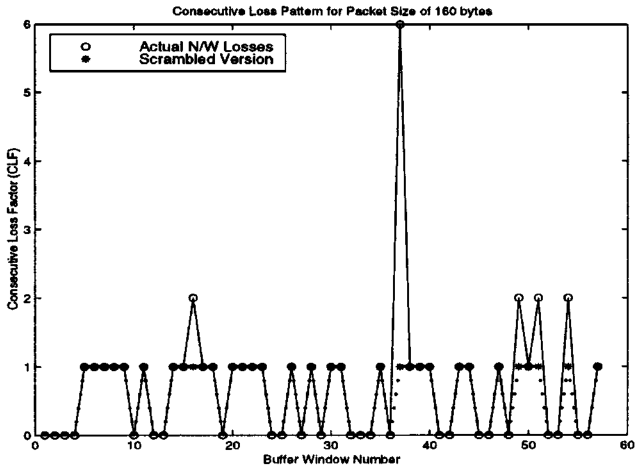

Fig. 9. Performance of our protocol for a real-time application such as Internet Phone.

Fig. 8 shows the result. As can be seen from the figure, our scheme does quite well in smoothing network consecutive losses. Due to space constraints, we show the performance with respect to clip 1. Similar resultes were obtained for clip 2. Most of the time CLF is well below and also within tolerable perceptual limits (see Section II-A).

B. Experiment: Using Data From a Real Time Application Like Internet Phone

The data was collected for an *Internet Voice or Voice on Networks* (VON) application. The server is textitvermouth.ee.umanitoba.ca (Canada) and the client is rawana.cs.umn.edu (Minnesota, USA). *vermouth* and *rawana* are both SUN UltraSparc 1, running Solaris V2.6 and V2.5 respectively. Each host is on a 10 Mb/s Ethernet (LAN). The transmission is over the Internet and the data set was collected on a Saturday, from 10 am to 2 pm. The data was collected for voice packets of sizes 160 and 480 bytes. We provide the graphs for voice packets of size 160 bytes while the performance of the other is similar.

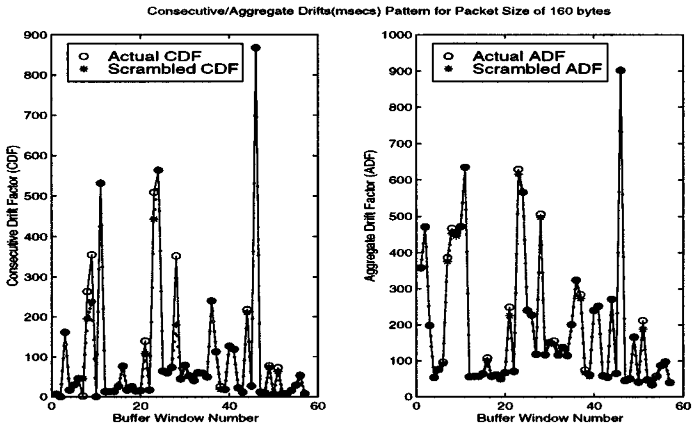

Fig. 10. Drift factors.

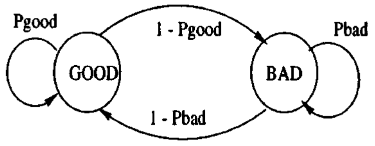

Fig. 11. Two-state Markov model of network loss.

As can be seen from Fig. 9, the actual CLFs due to network losses are varying while the CLF based on our protocol is always lower (in this case $\mathrm{CLF} = 1$, implying no consecutive losses).

As we have mentioned earlier, our scheme does not induce new drifts. To illustrate that point, we plotted the drifts (ADF, CDF) of the regular scheme versus our scheme, as shown in Fig. 10. As can be seen, there are no significant changes in drifts. Note that, timing drifts occur because of delay of frames delivered to the client from the network and/or the ability of clients to schedule events at fine time slots, depending on operating system load, etc. Timing drifts arising due to network delays translates to losses if drifts are longer, and hence is taken care of in our scheme. Drifts due to load at the client side can be controlled if the client has the ability to schedule events in a fine grained manner, and is not addressed in this paper.

C. Simulation

1) Simulation Model: Network loss pattern is modeled by a two-state Markov model as shown in Fig. 11.

The two states are *good* (successful) state and *bad* (lossy) state. Since networks loose packets in burst, once in the good state, the model remains there with probability P_{good}. Once it switches to the bad state with probability $1 - P_{\mathrm{good}}$, it remains there with probability P_{bad}. It switches back to the good state with probability $1 - P_{\mathrm{bad}}$. Essentially, the regular MPEG transmission and our transmission differ only in the order of frames being sent in the sender's buffer window of $m = wF_{\mathrm{GOP}}$ LDUs. In this experiment, the data was taken from the MPEG trace of Jurassic Park, which can be obtained from ftp://ftp.cs.umn.edu/users/varadara/Traces. Jurassic Park MPEG clip has GOPs of $F_{\mathrm{GOP}} = 12$ frames each. Note that the simulation was conducted for fixed bandwidth (at the specified peak) and a fixed delay. The only variations are the probabilities of the network packet losses (P_{good} and P_{bad}),

TABLE V
SUMMARY OF SIMULATION RESULTS

Sec.	Varied params	Unscrambled CLF Mean	Unscrambled CLF Devn	Scrambled CLF Mean	Scrambled CLF Devn	Mean CLF redn. %
??	$Pbad = .6$	1.71	0.92	1.46	0.56	15%
	$Pbad = .7$	1.63	0.85	1.56	0.79	4%
??	1.2 Mbps	1.71	0.92	1.46	0.56	15%
	.7 Mbps	2.96	2.77	2.21	2.15	25%
??	1 sec.	1.71	0.92	1.46	0.56	15%
	3.5 sec.	2.04	0.62	1.84	0.49	10%

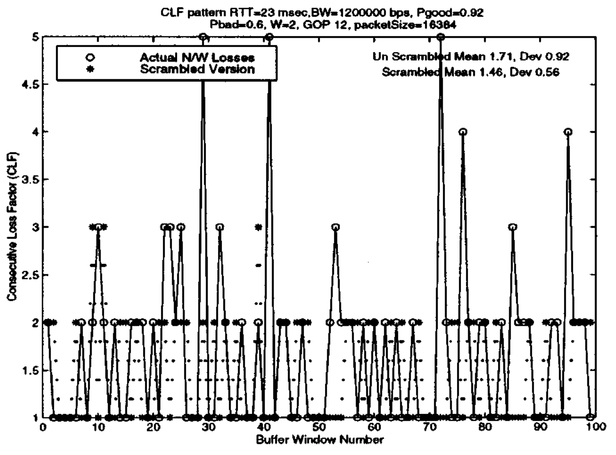

Fig. 12. Experiments on impact of network losses.

which are random derivates drawn from a uniform distribution in the interval $[0, 1)$. The network is initially in the good state.

As bandwidth, network loss, and buffer size are the main parameters which affect CLF, we measured the impact of these parameters separately. A set of other parameters are chosen to be at their typical values. Frames are broken up into packets of size $packetSize = 2$ kbytes. Round trip delay time is 23 ms. Probability that the network stays in the good state $P_{\rm good}$ is 0.92. We summarize our simulation result in Table V.

2) Impact of Network Loss on CLF: In our model, network loss is determined by the probability $P_{\rm bad}$ introduced earlier. The closer to $1 P_{\rm bad}$ is, the worse the network loss is. In the first two experiments, *bandwidth* is fixed at 1.2 Mb/s, buffer size is $W = 2$ GOPs, and $P_{\rm bad}$ is either 0.6 or 0.7. Fig. 12 shows the results for $P_{\rm bad} = 0.6$ while Table V shows the comparison. As can be seen, the error spreading scheme has been able to reduce both the mean and deviation of CLF over 100 buffer windows. The net effect is that better perceptual quality was achieved. The small reduction when $P_{\rm bad} = .7$ is rather surprising. However, due to the predictive nature of Eq. (1), it is expected that the protocol will occasionally not predict very accurately the next window's network loss. This phenomenon must happen to all protocols that use equations similar to (1), such as TCP.

3) Impact of Available Bandwidth on CLF: The next two experiments show that our scheme also perform better with different available bandwidth. Her, buffer size is kept at two GOPs, $P_{\rm bad}$ is 0.6, and bandwidth is varied from 700 kb/s to 1.2 Mb/s. Fig. 13 shows the results of the experiment with bandwidth 700 kb/s while Table V shows the comparison. Just as in the previous case, both the mean and standard deviation of CLF are improved. Moreover, our scheme often keeps CLF at or below

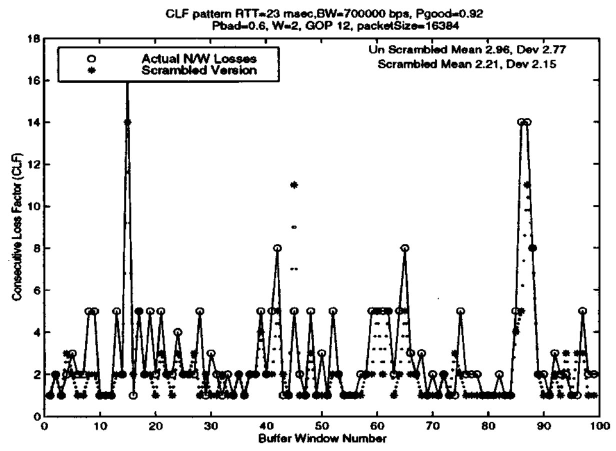

Fig. 13. Experiments on impact of available bandwidth.

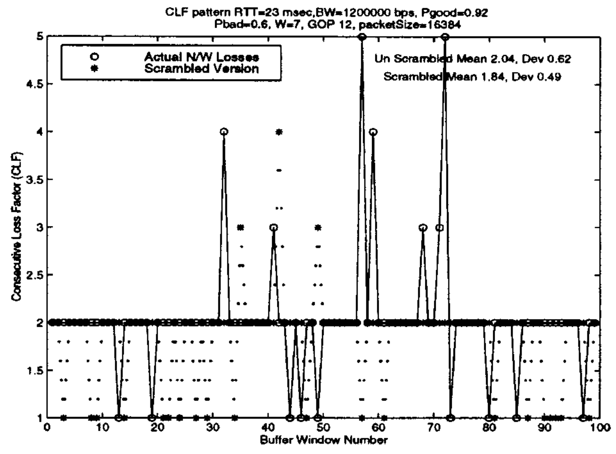

Fig. 14. Experiments on impact of buffer size.

2 which is the threshold for a perceptually acceptable video stream.

4) Impact of Buffer Size and Startup Delay on CLF: In the last two experiments, we vary W—the number of GOPs in the server's buffer. W is either 2 or 7, i.e., the initial delay time is either 1 or 3.5 s. Both values are acceptable in most practical situations. Network loss probability $P_{\rm bad} = 0.6$ and bandwidth is 1.2 Mb/s. Results of the experiment with $W = 7$ is presented in Fig. 14 while Table V shows the comparison. Again, both mean and deviation of CLF are better. This consistency proves a strong point that error spreading scales well in various scenarios.

5) Impact of Nonbursty Network Losses on CLF: When our fundamental assumption fails, namely the network losses are not bursty, it is natural to ask how our scheme behaves. In principle, we could observe a nonbursty network loss pattern, which increases CLFs when permutation scheme is applied.

The protocol can be easily modified to take into account this scenario, by measuring the burstiness of network losses. As soon as the predicted loss falls below a certain threshold of burstiness, no permutation needs to be applied. This slight modification does not lie on our line of reasoning, thus although it is easily implemented, we omit it in our protocol description.

In this section we measure our permutation scheme's performance **without** adjusting to the bursty level of network loss. We conducted two experiments when $P_{\rm bad}$ is 0.1 and 0.3. Low $P_{\rm bad}$

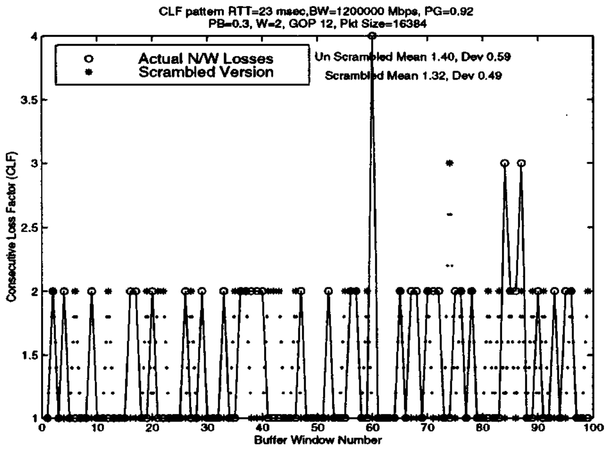

Fig. 15. Experiments on impact of nonbursty network loss.

is equivalent to low burstiness in network loss. Fig. 15 shows the case $P_{\text{bad}} = 0.3$ and $P_{\text{bad}} = 0.1$ has similar results. The experiment show that statistically the scrambling scheme is definitely not worse than its nonscrambling counterpart even if the main assumption fails.

VII. Conclusion

In this paper, we have addressed the problem of handling bursty losses in continuous media transmission. We formulated the problem in terms of a number of parameters including user QoS requirements, sender resource availability, and network loss behavior. We introduced the concept of *Error Spreading* which is a transformation technique that takes the input sequence of packets and permutes them before transmission. We then determined the bounds on achievable CLF using this kind of techniques for independent streams like MJPEG in a deterministic scenario. We also provided an algorithm to generate an appropriate permutation satisfying the above bounds. Next, we extended this idea to handle streams with inter-frame dependencies such as MPEG. Streams with inter-frame dependencies were modeled as partially ordered sets and the independent stream was shown to be just a special case of this. In addition, we presented a protocol for a nondeterministic network scenario and showed the orthogonal nature of our scheme with other CM error handling protocols. Finally, we validated the effectiveness of our scheme through a set of simulations and experiments over the Internet. The results indicated that the scheme makes the transmitted streams more acceptable from a perceptual viewpoint [27].

Appendix

In this Appendix, we shall state the two Lemmas concerning two main cases of Theorem 3.2. Lemma A.1 presents the case where $p \leq (m/2)$. A formal proof of this lemma is provided. Lemma A.2 deals with the case where $(m/2) < p < m$. Due to lack of space, this lemma is stated without proof. Interested readers are referred to [11] for more details.

Lemma A.1: If $0 < p \leq (m/2)$ then $k_0 = 1$

Proof: Since $p > 0$, we have $k_0 \geq 1$. So, to prove $k_0 = 1$ it is sufficient to specify a permutation π on $S = \{1, 2, 3, \ldots m\}$ so that $C^\pi = 1$. To avoid possible confusion, we would like to point out that for any $\pi \in S_m, \pi(i)$ specifies the position of i in the permuted sequence where i is sent to, while π_i is the number at position i in one line notation of π.

We consider two cases based on the parity of m as follows.

- **Case 1**: m **is odd**
  Let $m = 2m' + 1, p' = \min\{j, j \geq p \wedge \gcd(m, j) = 1\}$. Since $p \leq (m/2)$, we have $p \leq m'$. Moreover, $\gcd(m, m') = 1$, hence $p' \leq m' < (m/2)$. We now construct a permutation π such that $C^\pi = 1$. Let π be defined as follows.

$$\pi(i) = ((i - 1) \cdot p' \bmod m) + 1, \; 1 \leq i \leq m \tag{2}$$

  We first need to prove that π is indeed a permutation. As $\gcd(m, p') = 1, p'$ generates the group $\mathbb{Z}_m$ of integers modulo $m(\mathbb{Z}_m = \{0, 1, \ldots m - 1\})$. Thus the sets $\{0, 1, \ldots m - 1\}$ and $\{0, p', 2p' \bmod m, \ldots (m - 1)p' \bmod m\}$ are identical. So (2) defines a valid $\pi \in S_m$.
  Secondly, we need show that $C^\pi = 1$. Let $\pi = \pi_1, \pi_2, \ldots \pi_m$ in one line notation. If $C^\pi \geq 2$ then there exists i and $j, 1 \leq i < j \leq m$ such that **either** $|\pi_i - \pi_j| = 1$ and both π_i and π_j belong to the same sliding window W_k^π for some $1 \leq k \leq m - p + 1$ **or** $\{\pi_i, \pi_j\} = \{1, m\}$ and both π_i and π_j belong to the same sliding window W_k^π for some $m - p + 1 < k \leq m$.
  Note that by definition of π, we have $i = ((\pi_i - 1).p' \bmod m) + 1$ and $j = ((\pi_j - 1).p' \bmod m) + 1$, thus,

$$j - i = (\pi_j - \pi_i).p' \bmod m \tag{3}$$

  — If $|\pi_i - \pi_j| = 1$ and both π_i and π_j belong to the same W_k^π for some $1 \leq k \leq m - p + 1$, then we must have $j - i < p$. Moreover, $|\pi_i - \pi_j| = 1$ and (3) imply $j - i$ is either p' or $m - p'$. As stated earlier, $p \leq p' < m/2$, so $m - p' > p' \geq p$, making $j - i < p$ impossible.
  — Otherwise, suppose $\{\pi_i, \pi_j\} = \{1, m\}$ and both π_i and π_j belong to the same sliding window W_k^π for some $m - p + 1 < k \leq m$. Notice that we must have $m - j + i < p$ for both π_i and π_j to be in W_k^π. Moreover, $\{\pi_i, \pi_j\} = \{1, m\}$ and (3) imply that $j - i$ is either p' or $m - p'$. So $m - j + i \in \{p', m - p'\}$, and similar to the previous case, this makes $m - j + i < p$ impossible
  In sum, we have just shown that $c_i^\pi = 1, \forall 1 \leq i \leq m$, so $C^\pi = 1$. Notice that the choice of p' could have been any integer between p and $m/2$, as long as $\gcd(m, p') = 1$. In particular, $p' = m'$ clearly works. However, we have chosen p' to be the minimum of these because we would like the permuted sequence to be spread out in a "better manner" (see [11] for explanations).
- **Case 2**: m **is even**
  If $\{j, p \leq j < (m/2) \wedge \gcd(m, j) = 1\} \neq \emptyset$ then we can just use exactly the same permutation as when m is odd. If the set is empty, let $m = 2p' \geq 2p$, so $p' \geq p$. As the previous case, we seek a permutation π so that $C^\pi = 1$. Let π be defined as follows.

$$\pi(i) = p' \cdot (i \bmod 2) + \left\lceil \frac{i}{2} \right\rceil, \quad 1 \leq i \leq m \tag{4}$$

If i is even, then $\pi(i) = (i/2)$, namely all the even numbers will be placed from the first position to the $(m/2)$th position in increasing order. When i is odd, $\pi(i) = p' + ((i+1)/2)$, so all the odd numbers will be placed from the $((m/2)+1)$st position to the mth position. It is clear from the above observation that π is a proper permutation on S.

We are left to prove that $C^{\pi} = 1$. Write $\pi = (\pi_1, \pi_2, \ldots \pi_m)$ in one line notation. Firstly, notice that for any $1 \leq i < j \leq m$, we have $i = p'.(\pi_i \bmod 2) + \lceil(\pi_i)/(2)\rceil$ and $j = p' \cdot (\pi_j \bmod 2) + \lceil(\pi_j)/(2)\rceil$.

So,

$$j - i = p' \cdot ((\pi_j - \pi_i) \bmod 2) + \left\lceil \frac{\pi_j}{2} \right\rceil - \left\lceil \frac{\pi_i}{2} \right\rceil \tag{5}$$

Similar to case 1, if $C^{\pi} \geq 2$ then we consider two subcases:

— If $|\pi_i - \pi_j| = 1$ and both π_i and π_j belong to the same W_k^{π} for some $1 \leq k \leq m - p + 1$ then we must have $j - i < p$. Moreover, since $|\pi_i - \pi_j| = 1$ and $j > i$, it must be the case that π_j is odd and π_i is even. Combining with (5), we have

$$j - i = p' + \left\lceil \frac{\pi_j}{2} \right\rceil - \left\lceil \frac{\pi_i}{2} \right\rceil \geq p' \geq p$$

This makes $j - i < p$ impossible.

— Otherwise, if $\{\pi_i, \pi_j\} = \{1, m\}$ and both π_i and π_j belong to the same W_k^{π} for some $m - p + 1 < k \leq m$, then we must have $m - j + i < p$ for both π_i and π_j to be in W_k^{π}. By (4), $\pi(1) = p' + 1$ and $\pi(m) = p'$, so it must be the case that $j = p' + 1$ and $i = p'$. Thus, $m - j + i = m - 1 \geq p' \geq p$, contradiction !

Lemma A.2: If $(m/2) < p < m$ then $k_0 = \lfloor (p/(m - p + 1)) \rfloor + 1$

Proof: See [11] for details. Although the previous lemma gives the intuition behind our result, the proof of this lemma is considerably more complex. □

Example A.3: Let $m = 17, p = 9$, then $k_0 = 2$, and $\pi = (16\ 13\ 10\ 7\ 4\ 1\ 15\ 12\ 9\ 6\ 3\ 17\ 14\ 11\ 8\ 5\ 2)$.

Example A.4: Let $m = 17, p = 12$, then $k_0 = 3$, and $\pi = (16\ 12\ 8\ 4\ 17\ 15\ 13\ 11\ 9\ 7\ 5\ 3\ 1\ 14\ 10\ 6\ 2)$.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for many suggestions, which have helped improve the presentation of this paper significantly. They would also like to thank D. Wijesekera, D. Su, and R. Harinath, Computer Science Department, University of Minnesota, and T. C. Nguyen of the Department of Computer Science, University of Wisconsin, Milwaukee.

REFERENCES

[1] A. Albanese, J. Blomer, J. Edmonds M. Luby, and M. Sudan, "Priority encoding transmission," in *Proc. 35th Annu. Symp. Foundations of Computer Sciences*, Santa Fe, NM, Nov. 1994, pp. 604–612.
[2] J. C. Bolot and A. V. Garcia, "The case for FEC-based error control for packet audio in the Internet," Multimedia Syst., to be published.
[3] J. C. Bolot and T. Turletti, "Experience with control mechanisms for packet video," *ACM Commun. Rev.*, vol. 28, no. 1, 1998.
[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. New York: McGraw Hill, 1990.
[5] W. C. Feng, F. Jahanian, and S. Sechrest, "An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video," *Multimedia Syst. J.*, vol. 5, no. 5, pp. 297–309, 1996.
[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.
[7] R. Hasimoto-Beltran and A. Khokhar, "Pixel level interleaving schemes for robust image communication," in *17th IEEE Symp. Reliable Distributed Systems*, 1998, pp. 455–460.
[8] W. Luo and M. El Zarki, "Analysis of error concealment schemes for MPEG-2 video transmission over ATM networks," in *SPIE/IEEE Visual Communications and Image Processing*, vol. 2501, Taiwan, May 1995, pp. 1358–1368.
[9] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM SIGCOMM*, 1996, pp. 117–130.
[10] H. Q. Ngo, S. Varadarajan, and J. Srivastava, "Error spreading: Reducing bursty errors in continuous media streaming," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems (ICMCS'99)*, vol. 1, 1999, pp. 314–319.
[11] ——, "On achieving lower consecutive losses for continuous media streams," Dept. of Computer Science, Univ. of Minnesota, Minneapolis, MN, Tech. Rep. TR99-005, 1999.
[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical evaluation," presented at the ACM SIGCOMM'98, Vancouver, CA, Sept. 1998.
[13] V. Paxson, "End-to-end Internet packet dynamics," presented at the ACM SIGCOMM'97, Cannes, France, 1997.
[14] S. V. Raghavan, S. K. Tripathi, and B. Prabhakaran, "On QoS parameters and services for multimedia applications," Dept. of Computer Science, Univ. of Maryland, College Park, MD, Tech. Rep. 3167, 1994.
[15] S. Ramanathan and P. Venkat Rangan, "Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems," *Computer J.*, vol. 36, no. 1, pp. 19–33, 1993.
[16] I. Rhee and S. R. Joshi, "Error recovery using FEC and retransmission for interactive video transmission," Dept. of Computer Science, North Carolina State Univ., Tech. Rep. June, TR-98-12, 1998.
[17] H. Schulzrinne, S. Casner, R. Frederick, and S. McCanne, RTP: A transport protocol for real-time applications, in RFC 1889, 1994.
[18] S. Sedigh, J. Joshi, A. Bashandy, and A. Ghafoor, "Evaluation of filtering mechanisms for MPEG video communications," in *17th IEEE Symp. Reliable Distributed Systems*, 1998, pp. 449–454.
[19] B. Smith. Cyclic-UDP: A priority driven best-effort protocol. [Online]. Available: http://www.cs.cornell.edu/Info/Faculty/Brian_Smith.html/Publications
[20] B. Smith, L. Rowe, and S. Yen, "A TCL/TK continuous media player," presented at the TCL 1993 Workshop, June 1993.
[21] B. C. Smith, "Implementation techniques for continuous media systems and applications," Ph.D. dissertation, Univ. of California, Berkeley, CA, 1994.
[22] R. P. Stanley, Enumerative combinatorics, Cambridge, U.K., vol. 1, 1997.
[23] R. Steinmetz and G. Blakowski, "A media synchronization survey: Reference model, specification and case studies," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 5–35, Jan. 1996.
[24] S. Varadarajan, H. Q. Ngo, and J. Srivastava, "An adaptive, perception-driven error spreading scheme in continuous media streaming," in *Proc. 20th IEEE Int. Conf. Distributed Computing Systems*, Apr. 2000, pp. 475–483.
[25] B. W. Wah and D. Lin, "Transformation-based reconstruction for audio transmissions over the Internet," in *17th IEEE Symp. Reliable Distributed Systems*, 1998, pp. 211–217.
[26] D. Wijesekera and J. Srivastava, "Quality of service (QoS) metrics for continuous media," *Multimedia Tools Applicat.*, vol. 3, no. 1, pp. 127–166, Sept. 1996.
[27] D. Wijesekera, J. Srivastava, A. Nerode, and M. Foresti, "Experimental evaluation of loss perception in continuous media," *Multimedia Syst.*, vol. 7, no. 6, pp. 486–499, 1999.
[28] J. H. Yao, Y. M. Chen, and T. Verma, "MPEG-based audio-on-demand experiment for the Internet," presented at the Internetworking'96, Nara, Japan, 1996.
[29] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Network*, vol. 5, pp. 8–18, 1993.

[30] Z. L. Zhang, S. Nelakuditi, R. Aggarwal, and R. P. Tsang, "Efficient selective frame-discard algorithms for stored video delivery across resource constrained networks," in *Proc. IEEE INFOCOM*, 1999, pp. 472–479.

Srivatsan Varadarajan (S'98) received the B.S. degree from Birla Institute of Technology and Science (BITS), Pilani, India, and the M.S. degree in computer science from the University of Minnesota, Minneapolis. He is currently working toward the M.S. degree in mathematics and the Ph.D. degree in computer science at the University of Minnesota.

His research interests are in QoS enabled multimedia services over the Internet and mobile ad-hoc wireless environments.

Mr. Varadarajan is a student member of the Association for Computing Machinery and the AMA.

Hung Q. Ngo received the B.S. degree in computer engineering from the Ho Chi Minh City University of Technology, Vietnam, and the M.Sc. degree in mathematics and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, State University of New York at Buffalo, where he does research on the theory of interconnection networks and combinatorial group testing.

Jaideep Srivastava (S'85–M'88–SM'99) received the B.S. degree from the Indian Institute of Technology, Kanpur, India, in 1983, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1985 and 1988, respectively.

Since 1988, he has been on the faculty of the University of Minnesota, Minneapolis, where he is a tenured Professor. For over 15 years, he has been active as a researcher, educator, consultant, and invited speaker, in the areas of data mining, databases, artificial intelligence, and multimedia. He has established and led a database and multimedia research laboratory, which has graduated 16 Ph.D. and 37 M.S. students, and he has published over 135 papers in journals and conferences. Throughout his career, he has had an active collaboration with the industry, both for collaborative research and technology transfer. Between 1999 and 2001, he was on leave from the University of Minnesota, during which period he spent time at Amazon.com (http://www.amazon.com) as the Chief Data Mining Architect, at Yodlee Inc. (http://www.yodlee.com) as Director—Data Analytics, and at Chingari Inc. (http://www.chingari.com) as the Chief Technology Officer. This wide-ranging industry experience has provided him with a unique perspective on the application of various computer science technologies in the Internet economy. He is an often-invited participant in technical as well as technology strategy forums. He has given more than a hundred talks in various industry, academic, and government forums. He has served on the program committee of a number of conferences, and is on the editorial board of various journals. The federal government has solicited his opinion on computer science research as an expert witness. He has also served in an advisory role to the governments of India and Chile on various software technologies.

Dr. Srivastava is a member of the Association for Computing Machinery.