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

Srivatsan Varadarajan, Hung Q. Ngo, and Jaideep Srivastava
Department of Computer Science, University of Minnesota,
200 Union street, EE/CS Building, room 4-192, Minneapolis, MN 55455
e-mail: {varadara, hngo, srivasta}@cs.umn.edu,

Abstract

With the growing popularity of the Internet, there is increasing interest in using it for audio and video transmission. Periodic network overloads, leading to bursty packet losses, have always been a key problem for network researchers. In a long-haul, heterogeneous network like the Internet, handling such an error becomes especially difficult. Perceptual studies of audio and video viewing have shown that bursty losses have the most annoying effect on people, and hence are critical issues to be addressed for applications such as Internet phone, video conferencing, distance learning, etc. Classical error handling techniques have focused on applications like FTP, and are geared towards 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 takes the input sequence of packets (from an audio or video stream) and scrambles its packets 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. Our error spreading idea deals with either cases where the stream has or does not have inter-frame dependencies. Perceptual studies have shown that users are much more tolerant to a uniformly distributed loss of low magnitude. We next describe a continuous media transmission protocol based on this idea. We also show that our protocol can be used complementary to other error handling protocols. Lastly, we validate its performance through a series of experiments and simulations.
Keywords: multimedia, network bursty error, permutation scheme.

1 Introduction

Due to the phenomenal growth of multimedia systems and their applications, there have been numerous research efforts directed at providing a *continuous media (CM)* service over varying types of networks. With the boom of the Internet, continuous media like audio and video are using the Internet as the principal medium for transmission. However, the Internet provides a *single class best effort* service, and does not provide any sort of guarantees [1]. 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 [2, 3], leading to significant bursty errors [4].This bursty loss behavior has been shown to arise from the drop-tail queuing discipline adopted in many Internet routers [5]. This problem may be reduced if RED gateways are used, wherein the probability that the gateway notifies a particular connection to reduce its window is roughly proportional to that connection's share of the bandwidth through the gateway [6]. Nevertheless since drop-tail queuing discipline is still adopted in many routers [5], bursty network errors have to still be reconciled with.

Handling bursty errors has always been problematic, especially since no good models exist for its prediction. On the other hand, most applications do not tolerate bursty error, making it imperative that they be handled in a good manner. Perceptual studies on continuous media viewing have shown that user dissatisfaction rises dramatically beyond a certain threshold of bursty error [7, 8, 9]. This is especially so for audio, where the threshold is quite small, and hence this issue is quite pressing for applications like the Internet phone. These observations point quite solidly to the need for development of efficient mechanisms to control bursty errors in continuous media streaming through networks. Redundancy is the key to handling packet loss/damage in standard communication protocols. There are two main classes of error handling schemes, namely the *reactive* schemes and the *proactive* schemes. Reactive schemes respond by taking some action once transmission error has been detected, while pro-active schemes take some action in advance to avoid errors . A protocol such as TCP is reactive since the receiver sends a feedback to the sender upon detecting an error, in response to which the receiver will transmit the lost packet. The reaction can be initiated by the source or the sink. Source initiated reaction occurs in schemes based on feedback combined with retransmission like [1, 10, 11]. The feedback control can be based on stream rate [12, 13], bandwidth [14], loss/delay [1] and a wide variety of network QoS parameters [15, 16, 17]. Client initiated reaction occurs in reconstruction based schemes like [18, 19]. Coding data in an error correcting manner before transmission is a pro-active scheme where any packet corruption can be handled because of the coding scheme ([20] and *Forward Error Correction Codes* [21, 22]). There exist hybrid schemes which use both forms of redundancy [23, 24]. Another technique that is called *error concealment* [25, 26], 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 [27, 28]. Services like RTP/RSVP require that some resource allocation and/or reservation mechanism be provided by the network [1].

Recent work ([29, 30]) has proposed schemes where the overall characteristics of the data being transmitted can be used to control the transmission error. For e.g., [30] has proposed selectively dropping video frames on the sender side, based on a *cost-benefit analysis* which takes into account the desired Quality of Service (QoS). This is quite effective in a LAN (senders are known and cooperative) or the Internet using RED gateways where during congestion, the probability that the gateway notifies a particular connection to reduce its window is roughly proportional to that connection's share of the bandwidth through the gateway [6]. This scheme may not be directly applicable yet as long as drop-tail queuing discipline is still common.

In this paper we propose a new type of scheme for handling bursty errors, which we call *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 continuous media viewing [7, 8, 9] has shown that a reasonable amount of overall error is acceptable, *as long as it is spread out, and not concentrated in spots*. A similar approach have been taken by [4]. However, the impact of packet scrambling on user perception or system resources was not investigated. In [31, 32] we had established precisely the relationship between buffer requirement and user perceived quality in a bounded bursty network error scenario . A key advantage of this approach is that it requires no additional bandwidth for the spreading of errors.

The error spreading scheme proposed in [31, 32] works only for streams with no inter frame dependency, e.g. MJPEG video, audio etc. In [33] we extend the above idea for streams with inter frame dependency, e.g. MPEG video. Further, we also show that the error spreading approach is complementary to other error handling schemes, and by using them in conjunction (at the expense of extra bandwidth of course), user perceived discontinuity can be further reduced. This paper is the consolidated work from [31, 32, 33].

In this paper we make several contributions. First, we formulate the problem of error handling in continuous media 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 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 continuous media 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 2 formulates the problem and Section 3 presents a mathematical analysis of the bounded network error case. Section 4 presents the general solution to streams with inter-frame dependencies. Section 5 provides a protocol where the network error is unbounded and which can be plugged into other error handling schemes. Section 6 presents results of experiments and simulations which validate our claim. Finally, section 7 concludes the paper.

2 Background

This section briefly discusses the content based continuity QoS metrics introduced in [8]. Next we give motivations for our work. Finally, we define our problem based on the metrics introduced.

2.1 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 [34]). 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 [8]. Relevant issues arising out of rates and drifts [8] are discussed briefly in Section 5.3. Note also that we shall use the term LDU and frame interchangeably.

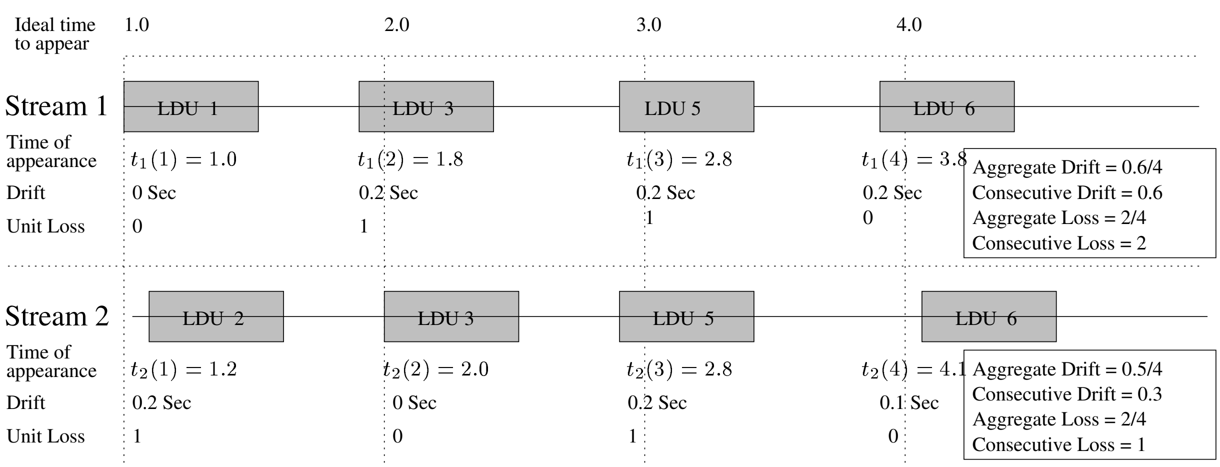


Figure 1: Two Example Streams used to Explain Metrics

Figure 1 is from [8]. 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 non-zero drifts from these ideals, over a given number of consecutive LDUs in a stream. For eg., first four LDUs of two example streams with their expected and actual times of appearance, are shown in Fig. 1. In the first example

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

	Frame sequence	Consecutive Loss/ Window Size
In order	01 02 03 04 05 06 $\boxed{07\ 08\ 09\ 10\ 11\ 12\ 13}$ 14 15 16 17	$7/17$
Permuted	01 06 11 16 04 09 $\boxed{14\ 02\ 07\ 12\ 17\ 05\ 10}$ 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 $\boxed{17}$	$1/17$

Table 1: An example of how the order of frames sent affects CLF

stream, the drifts are respectively 0.0, 0.2, 0.2 and 0.2 seconds; and accordingly it has an aggregate drift of 0.6 seconds per 4 time slots, and a non-zero consecutive drift of 0.6 seconds. In the second example stream, the largest consecutive non-zero drift is 0.3 seconds and the aggregate drift is 0.5 seconds per 4 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 over-load 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 [8].) The aggregate number of such unit losses is the *aggregate loss (ALF)* of a CM stream, while the largest consecutive non-zero loss is its *consecutive loss (CLF)*. In the example streams of Fig. 1, stream 1 has an aggregate loss of 2/4 and a 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 spread-out than those of stream 1. Note that the metrics already takes care of losses (both consecutive and aggregate) that arise due timing drifts.

In a user study [7] it has been determined that the tolerable value for consecutive losses were determined to be two frames. For audio this limit was about three frames.

2.2 Impact of Network Errors on QoS

One of the most important QoS parameters that affects the quality of a CM stream is the *Consecutive Loss Factor* [8] (CLF). Packet transmission on networks shows a bursty behavior, alternating between spurts of packet loss and packet delivery [4, 2, 3, 5]. This usually causes unacceptably high CLF from a user perception point of view. For example, suppose we sent 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 1. 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 1. With exactly the same bursty error once again consecutive frames are lost, except this time they are consecutive only in the permuted domain. In the original domain these are spread far apart. Clearly, if the 17 frames were sent in this order, we would have had a CLF of only $1/17$.

Table 1 summarizes our example by giving three sequences and their corresponding CLFs. The first sequence is the natural order of frames, the second is the permuted order, and the third is the un-permuted 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 is the need for a reasonable metric to compare permutations. We use the CLF metric introduced earlier, as it is directly related to human perception [8].

- 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 non-deterministic, and any permutation scheme must address this issue.

In this paper we address the issues listed above.

2.3 Problem Statement

We now formally state the problem.

Bursty Error Reduction Problem (BERD)

- **Objective :** to reduce the bursty error, i.e. CLF, to an 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 $S = \{1, 2, 3, \ldots, m\}$ 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 loss are at least m LDUs apart.

Figure 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, at or above which the buffer size should be to support k_0.

There are several points worth noticing. Firstly, we deal only with data streams that have no inter-frame dependency such as Motion-JPEG or uncompressed data streams (audio, video, sensor data, ...). The reason for this is that this allows us to consider every frame to be equally important; thus, we can permute the frames in any way we would like to. Secondly, 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 it is

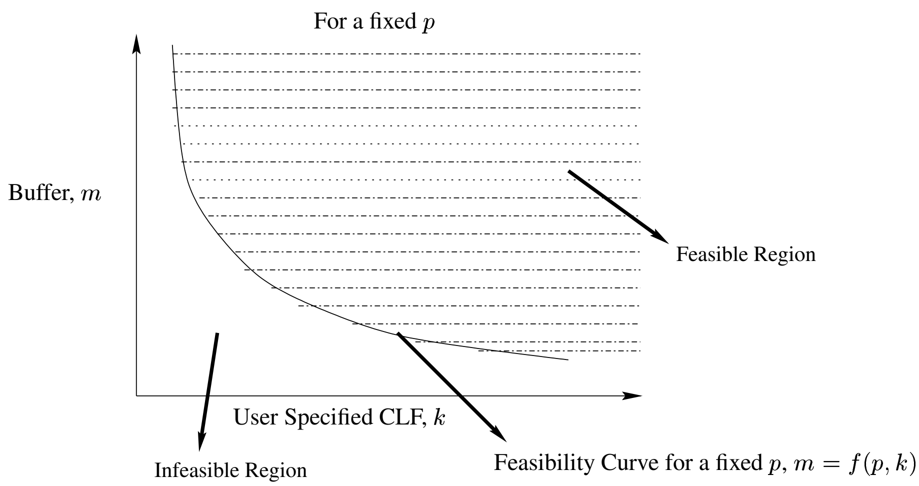


Figure 2: Part of Deterministic Solution Space

us who send the frames by breaking them up into equal size UDP packets or 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.

3 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 acting on $[m] = \{1, 2, \ldots m\}$. For any permutation $\pi \in S_m$, the sets $W_i^\pi = \{\pi_i, \pi_{i+1}, \ldots \pi_{i+p-1}\}, 1 \leq i \leq m$ are called the *sliding windows* of size p (of π), where the indices are calculated modulo m, then plus 1. Thus, when $1 \leq i \leq m - p + 1$, $W_i^\pi = \{\pi_i, \pi_{i+1}, \ldots \pi_{i+p-1}\}$, and when $m - p + 1 < i \leq m$, $W_i^\pi = \{\pi_i, \ldots, \pi_m, \pi_1, \ldots, \pi_{i+p-m-1}\}$

For any pair of integers k and l such that $1 \leq k < l \leq m$, 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.

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

Let $C^\pi = \max\{c_i^\pi | 1 \leq i \leq m\}$. 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 $1 \leq i \leq m - p + 1$, c_i^π is the maximum number of consecutive integers in W_i^π. While if $m - p + 1 < i < m$, c_i^π 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 $calulatePermutation(m, p)$ 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 $p \geq m$.

Theorem 3.2. *If p and m are both determined, then*

- $k_0 = 0$ *when* $p = 0$ *and* $k_0 = m$ *when* $p \geq m$
- $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$ *when* $0 < p < m$.

Proof. Since if $0 < p \leq \frac{m}{2}$ then $\left\lfloor \frac{p}{m-p+1} \right\rfloor = 0$, this is immediate 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 which generates permutation π with $C^\pi = k_0$ on input m and p. Notice that it takes only linear time.

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

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.

4 Modeling Streams with Inter-Frame Dependency

This section is organized as follows. Section 4.1 defines related concepts from Combinatorics, 4.2 discusses our solution for MPEG streams, and section 4.3 generalizes the analysis for any stream with inter-frame dependencies.

4.1 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 [35].

A *poset* (or *partially ordered set*) 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 $x \rightarrow z$.

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 are incomparable. An *antichain decomposition* of size w of P is a partition of P into 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 $\nexists z \in P, z \neq x, z \rightarrow x$. 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 [36]. Thus, a linear extension of a poset can be obtained by a topological sort algorithm.

4.2 Analysis of the MPEG case

Figure 3 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 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 figure 3, 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 *non-anchor 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.

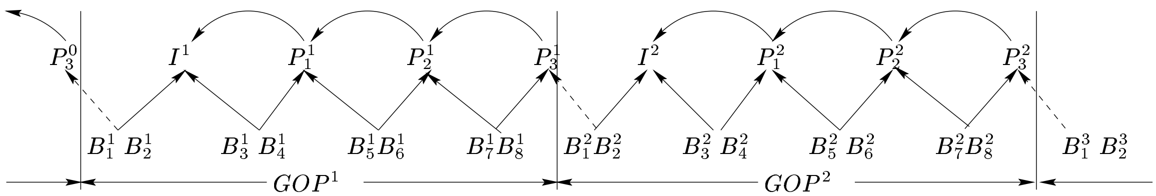


Figure 3: A Sample Pattern of MPEG Frame Dependency

CM streams with inter-frame dependency can be modeled as a poset P of frames, where for any two frames x and y, $x \to y$ 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 $\exists x \in P, x \to y$. 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 should be a linear extension of P where the anchor frames go first, since the non-anchor frames can not be reconstructed without the anchor frames.

To illustrate this, consider Figure 3 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 then replaces P_3^0, etc; 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 loss or server runs out of buffer cycle time. B_7^2, B_8^2, B_1^3, and B_2^3 are all considered to be lost since we need P_3^2 to reconstruct them. This causes a CLF of 5 – more than twice the perceptually tolerable level of CLF [7].

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 : $I^1, I^2, P_1^1, P_1^2, P_2^1, P_2^2, P_3^1, P_3^2$, 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_1^i's $1 \leq i \leq w$, permute P_2^i'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 by figure 4. Here, f is the permutation function generated by the $calculatePermutation$ algorithm.

Notice that the dependencies we are considering here are the 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

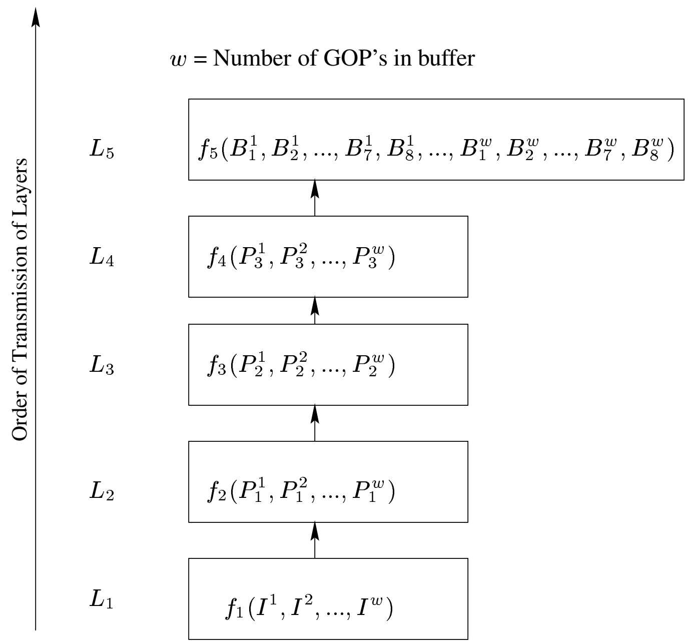


Figure 4: Layered Permutation Transmission Order for MPEG

[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 hierachical description, it is conceivable that the generalized solution presented in Section 4.3 could be applicable.

4.3 Generalized Solution

In general, given a CM stream whose inter-frame dependency is modeled by poset P, a general solution must answer the following questions: (a) which sets of frames can be permuted before transmission, and (b) which sets of frames form each layer in the proposed scheme. To answer (a), 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 (b), observe that **if** there exists an antichain decomposition $A_1, A_2, \ldots A_w$ of P such that $1 \leq i < j \leq w$, $\Rightarrow \forall x \in A_i, \forall y \in A_j,\ y \not\rightarrow x$ **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. Consequently, P being ranked automatically gives us the best antichain decomposition from which the layering can be derived.

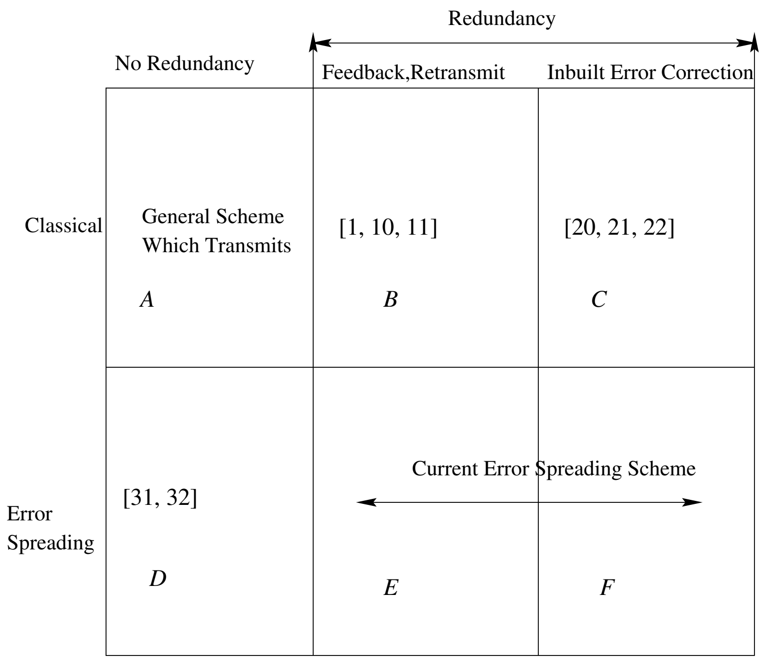


Figure 5: Classification of Error Handling Schemes

5 Error Spreading Protocol

Perceptual studies have established the tolerance levels for discontinuity in CM viewing [7, 8, 9]. Discontinuities beyond those levels have been shown to increase user dissatisfaction dramatically. 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 Figure 5. Our earlier work [32, 31] has shown that the scrambling scheme decreases CLF, the measure of 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. As can be seen from section 3, 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 block A and D are not very useful.

A variety of schemes have been proposed in the literature (block B and C in Figure 5) which use some form of redundancy to handle transmission failure in streams with inter-frame dependency. Reactive schemes are feedback/retransmission based (block B), while proactive schemes which uses forward error correction in the transmitted stream (block C). Additionally, there are hybrid schemes which are both proactive and reactive[23]. In Section 4 we provided a scrambling order for dependent streams like 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.

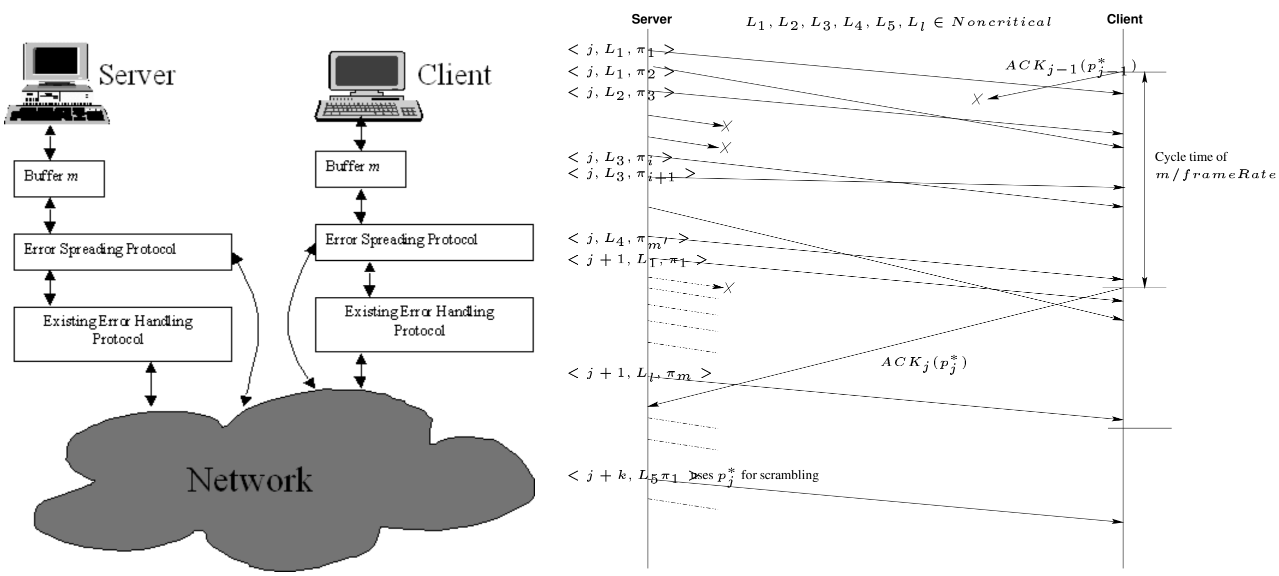
This section is organized as follows. Section 5.1 addresses the buffer requirement for the use of our protocol. Section 5.2 discusses how our protocol functions. Lastly, section 5.4 establishes the complementary nature of our protocol with other error handling schemes. The rest of this section uses MPEG as an illustration, though our technique can be applied to any dependent stream.

[2] See ISO/IEC 14496-1, Coding of audio-visual objects: systems, final draft international standard, ISO/IEC JTC1/SC29/WG11 N2501, October 1998.

GOP Size Info	Average (bits)	Deviation (bits)	Max (bits)	Min (bits)	Number of GOPs
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

Table 2: GOP data information for MPEG traces

5.1 Buffer Requirements



(a) *(b)*

Figure 6: (a) Protocol Setting and (b) a Protocol Illustration

As shown in Figure 6 (a), the server and client both require a buffer of size $m = w*F_{GOP}$, where F_{GOP}=Maximum Size of a GOP, and w = Number of GOPs, $w \geq 2$. Table 2 gives GOP statistics for different MPEG traces (traces can be found from "ftp://ftp.cs.umn.edu/users/varadara/Traces" or "ftp://gaia.cs.umass.edu/pub/zhzhang/Traces"). The data has traces with GOP size 15 (30 frames a second) as well as GOP size 12 (24 frames a second). Observe that the largest GOP size is 932710 bits or 113 Kbytes. Thus, $m = 2 * F_{GOP} = 2 * 113 = 226Kbytes$ is quite viable.

5.2 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 given of Figure 4, $l = 5$. A layer is *critical* if it contains anchor frames, on which other frames depend. Let the set of Critical Layers be *Critical* $= \{L_1, L_2, ..., L_q\}$ where $1 \leq q \leq l$. Hence, $Noncritical = L - Critical = \{L_{q+1}, ..., L_l\}$. In Figure 4, $Critical = \{L_1, L_2, L_3, L_4\}$ and $Noncritical = \{L_5\}$. Since the successful receipt of critical layers is important, the transmission order of layers is L_1, L_2, ..., L_l. In the proposed scrambling scheme, frames in L_i

are scrambled using the scrambling function $f_i()$ which is generated by the call $calculatePermutation(|L_i|, p)$; whose algorithmic details are given in Section 3. $|L_i|$ is the number of frames in L_i and p is the upper bound of network bursty loss within a window of $|L_i|$ LDUs.

Some CM systems use TCP/IP for communication [37]. However it has been shown in [38] that CM applications based on TCP are unstable when the real time bandwidth requirements are larger than the available bandwidth. Thus the proposed protocol uses the UDP communication model, as in [39, 40], 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 non-critical 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 $L_i \in Critical$) 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 non-critical 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 $CycleTime = m/frameRate$ to fill up the client's buffer (frame rate is typically 30 or 24 fps). 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 $L_i \in Noncritical$. It waits for $CycleTime$ and transmits the next estimated loss rate p_i for all layers $L_i \in Noncritical$ 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 $L_i \in Noncritical$ for the n^{th} window of m LDUs. Initially, when $n = 1$, the server assumes the average case where $p_{n,i} = \left\lfloor \frac{|L_i|}{2} \right\rfloor$. We use exponential averaging to estimate the next loss. Suppose we are currently at the n^{th} window, $p^*_{n,i}$ is determined by

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

We have picked $\alpha = \frac{1}{2}$, 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 6.

Figure 6 (b) illustrates an example of how client and server interact during transmission of layers $L_i \in Noncritical$. The transmission of layers $L_i \in Critical$ is easily seen as the server *does not require client feedback information* (when our protocol is used alone) and hence is not illustrated. In figure 6 (b), there is no indication of redundancy (retransmissions/error correcting code). $< j, L_3, \pi_i >$ is the time when the sender sends the i^{th} permuted frame of layer L_3 on j^{th} 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 k^{th} buffer window. Lastly, 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, the details of which are given in [32]. Also as shown earlier, our approach works for any encoding scheme, even though the illustrations are with MPEG.

5.3 Delay factors

Delay factors typically include startup delay and individual timing drifts (as defined in Section 2.1). 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. In Section 6.2, 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

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

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 2. Suppose we use a buffer of 2 GOPs with 15 frames per GOP. Further assume that our bandwidth is $10Mbps$ and $rtt = 23ms$, then the startup delay we need even for the largest GOP (Starwars, 932710 bits) is

$$\begin{aligned} d &= 2 \times \frac{1}{2} + 2 \times 932710/10^7 + 23/(2 \times 1000) \\ &\approx 1.2 \text{ sec.} \end{aligned}$$

The number for 4 GOPs is roughly doubled, i.e. 2.4 seconds, which is clearly viable for most purposes. For audio data, the startup delay is clearly a lot less and thus it is not of our concern here.

Notice that we do not use the buffer for prefetching and/or smoothing rates, etc. These mechanisms can be done in conjunction with our scheme and the manner in which this can be done is the subject matter of 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.

5.4 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 be built an *error spreading module* which uses a control channel for loss rate estimation, independent of any other error handling protocol (refer Figure 6 (a)). It is clear that any error handling protocol using retransmission or forward error correcting codes, e.g. [1, 10, 11, 15, 18, 21, 22, 23], or error concealment protocol [25, 26], 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 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 second.
- The sender needs to pick up the frames to be transmitted from the error spreading module's buffer .

	Frame sequence	Consecutive Loss/ Window Size
In order	01 02 03 $\boxed{\text{04 05 06 07 08}}$	$5/8$
IBO	01 05 03 $\boxed{\text{07 02 06 04 08}}$	$3/8$
Our order	01 04 07 $\boxed{\text{02 05 08 03 06}}$	$2/8$

Table 3: 8 frames ordering of IBO and one of the cases of our scrambled order

- The receiver informs the client side error spreading module of the received frames in 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)* [40].

We first briefly describe CMT and its CM delivery mechanism. CMT is an extensible system consisting of a three layered architecture. The topmost layer, referred to as the *application code layer*, consists of Tcl, Tk and Tcl-Dp [41] code. The next layer, consists of a CM object model, time and synchronization services, CM event services and storage/buffer services for CM streams. The bottom most *resource layer* consists of operating system and hardware services. The complete design of CMT is explained in detail in [40]. CMT envisions multimedia as streams flowing in and out of CM objects, in their journey between *sources* and *sinks*. These CM objects correspond to *services* that are used by CM streams, such as *segments*, that represent data sources stored in files. Similarly, *play* objects represent stream players, such as speakers and video displays. CMT provides distributed multimedia services in the sense that objects in a CM pipeline can sit on different locations, thereby requiring network transport services. The network twin objects *packet source (pktSrc)* and *packet destination (pktDest)* provide these services of sending and receiving streamed data.The Logical Time System (LTS) in the CMT Library layer provides a mechanism for applications to maintain a concept of where in time the application is, and how fast time is progressing. More than one LTS can exist in one application and different CM processes can be synchronized by using the same LTS. CMT ensures that the LTS progresses according to the *rate* specified by the application program.

A CMT application program specifies pipelines of CM streams flowing between CM objects in Tcl/Tk. The objects can be created within some *CM process*. These scripts are interpreted by a Tcl interpreter extended with Tcl-DP to provide distributed services. For an application that uses only one location, a single CM process is created with appropriate sources and sinks. Distributed applications have CM processes at each location, where objects belonging to that location are created within the corresponding process.

In particular, to deliver MPEG streams the cmFileSegment object is created at the sender. This object reads the file in, decodes it into separate frames, prioritizes and reorder the frames based on frame types and put them into a common buffer. Another object called pktSrc (packet source) pick up frames from the common buffer, decides which frames in the buffer to be sent using its estimated measure of various network parameters such as bandwidth and propagation delay. pktSrc can drop a set of low priority frames if it estimates that it can not deliver all of the frames in the buffer on time. 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 lost and time still allows. 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 3.Note that the 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.

	Frame size range	Median frame size	Mean frame size	Std deviation	Number of frames
Clip 1	5276-36364	9544	10845	4450.7	2607
Clip 2	5072-34408	10282	10916	3642.8	1736

Table 4: Summary data for the video clips

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 the LTS objects mentioned earlier. Losses of B frames occur only in the tail of the set of B frames because of the way CMT protocol works. It sends a bunch 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 3, 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 the better CLF. Further in our layered order, we use our ordering mechanism among the I and P frames ensuring more number of consecutive GOP's 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 4. Thus our model and approach can be extended to any type of continuous media and its delivery, reducing CLF even in pathological network scenario.

6 Experimental Evaluation

The following two sections presents the evaluation of our scheme in two scenarios. In section 6.1 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, 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* of CLF values are within the range of perceptual tolerance (see section 2.1). In section 6.3, we present the results of a detailed simulation based evaluation of our protocol using the layered permutation transmission order for MPEG described in section 4.2, and compare it against the usual MPEG transmission model.

6.1 Experiments

6.1.1 Video Experiment : Actual 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 a UltraSparc 1 (rawana.cs.umn.edu) in Computer Science department, University of Minnesota to another SunSparc (lombok.cs.uwm.edu) in Computer Science department, University of Wisconsin, Milwaukee [3]. The experiment was conducted at 9:45am when network traffic is expected to be average. Both clips have resolution 512×384. Other informations about the clips are summarized in Table 4. 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 (160.94.148.254)  2 ms  1 ms  1 ms
 2  tc8x.router.umn.edu (128.101.192.254)  23 ms  4 ms  3 ms
```

[3]Thanks to Mr. Thanh C. Nguyen at the Department of Computer Science, University of Wisconsin, Milwaukee for helping us in conducting this experiment

```
 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
```

Figure 7 shows the result. As can be seen from the figure, our scheme has done quite well smoothing network consecutive losses. In a few cases our CLF is 1 higher (clip 2) but that was due to rapid changes in network loss behavior and it is expected. Most of the time CLF is well below and also within tolerable perceptual limits (see section 2.1).

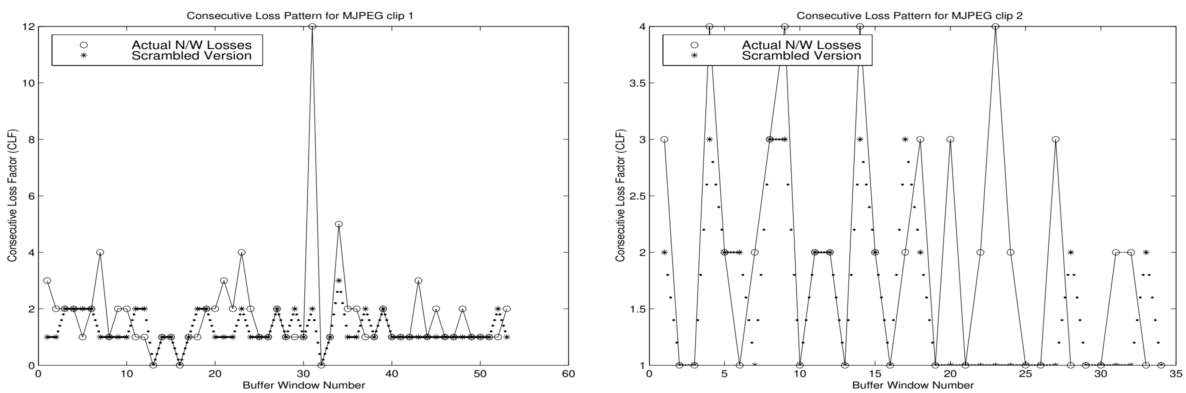


Figure 7: Performance of our protocol when transmitting video over long haul network

6.2 Experiment: Using data from a real time application like Internet Phone

The data [4] was collected for an *Internet Voice or Voice on Networks (VON)* application. The server is *vermouth.ee.umanitoba.ca (Canada)* and the Client is *rawana.cs.umn.edu (Minnesota, USA)*. *vermouth* and *rawana* is a SUN UltraSparc 1, running Solaris V2.6 and V2.5 respectively. Each host is on a 10 Mbps Ethernet (LAN). The transmission is over the Internet and the data set was collected on a Saturday, from 10 am to 2 pm. The two files presented here are of voice packets of sizes 160 and 480 bytes. As can be seen from figure 8 the actual CLF's of network losses are varying while the CLF based on our protocol always has lower CLF (in this case 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 Figure 9. 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. The timing drifts arising due to network delays translates to losses if drifts are longer, and hence is taken care of in our scheme. The 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 in the scope of this paper.

[4]Thanks to Mr. Difu Su, Computer Science Department, University of Minnesota, for providing us with the data set

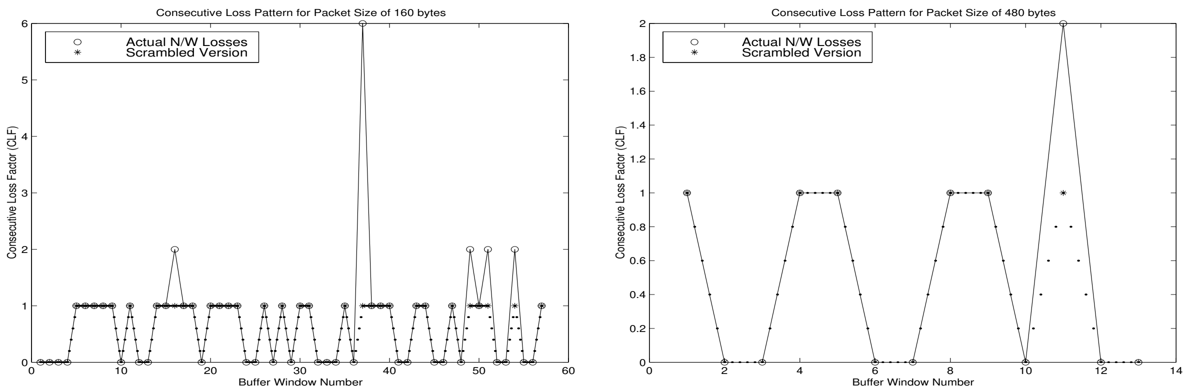


Figure 8: Performance of our protocol for a real-time application such as Internet Phone

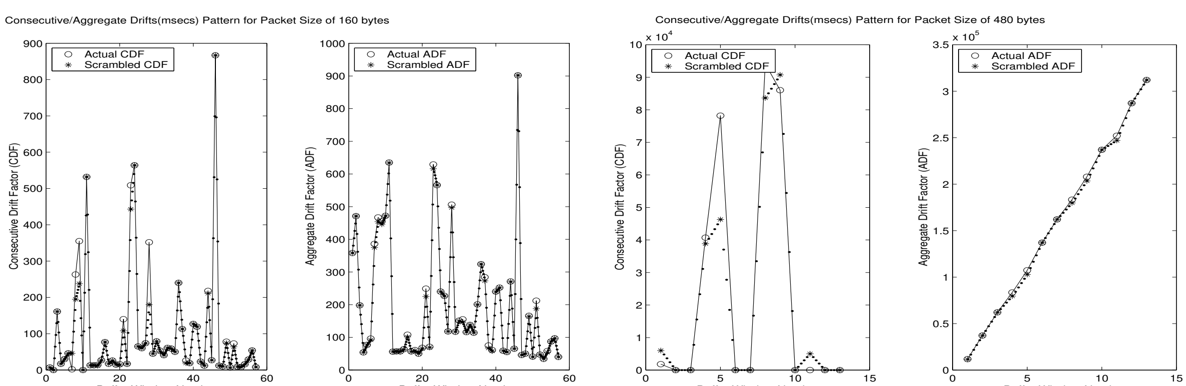


Figure 9: Drift factors

Impact of	Varied parameters	Unscrambled CLF		Scrambled CLF		Mean CLF reduction percentage
		Mean	Deviation	Mean	Deviation	
Network losses	$Pbad = .6$	1.71	0.92	1.46	0.56	15%
	$Pbad = .7$	1.63	0.85	1.56	0.79	4%
Available bandwidth	1.2 Mbps	1.71	0.92	1.46	0.56	15%
	.7 Mbps	2.96	2.77	2.21	2.15	25%
Initial delay and buffer size	1 sec.	1.71	0.92	1.46	0.56	15%
	3.5 sec.	2.04	0.62	1.84	0.49	10%

Table 5: Summary of simulation results

6.3 Simulation

6.3.1 Simulation Model

Network loss pattern is modeled by a two state Markov model as shown in Figure 10

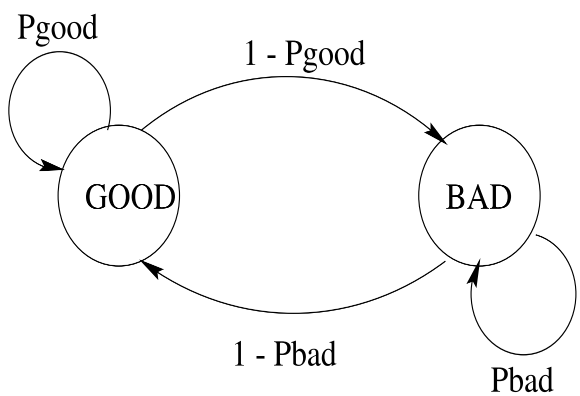


Figure 10: Two State Markov Model of Network Loss Pattern

The two states are *GOOD* (successful) state and *BAD* (lossy) state. Since networks lose packets in burst, once in the good state, the model remains there with probability $Pgood$. Once it switches to the bad state with probability $1 - Pgood$, it remains there with probability $Pbad$. It switches back to the good state with probability $1 - Pbad$. Essentially, the regular MPEG transmission and our transmission differ only in the order of frames being sent in the sender's buffer window of $m = wF_{GOP}$ 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" or "ftp://gaia.cs.umass.edu/pub/zhzhang/Traces". Jurassic Park MPEG clip has GOPs of $F_{GOP} = 12$ 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 ($Pgood$ and $Pbad$), 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 effect CLF, we measured the impacts of these parameters separately in the subsection followed. 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. The probability that the network stays in the good state $Pgood$ is 0.92. We summarize our simulation result in Table 5

6.3.2 Impact of Network Loss on CLF

In our model, network loss is determined by the probability $Pbad$ introduced earlier. The closer to 1 $Pbad$ is, the worse network loss is. In the first two experiments, $bandwidth$ is fixed at 1.2 Mbps, buffer size is $W = 2$ GOPs, and $Pbad$ is either 0.6 or 0.7. Figure 11 shows the results. 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 $Pbad = .7$ is rather surprising. However, due to the prediction nature of equation (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.

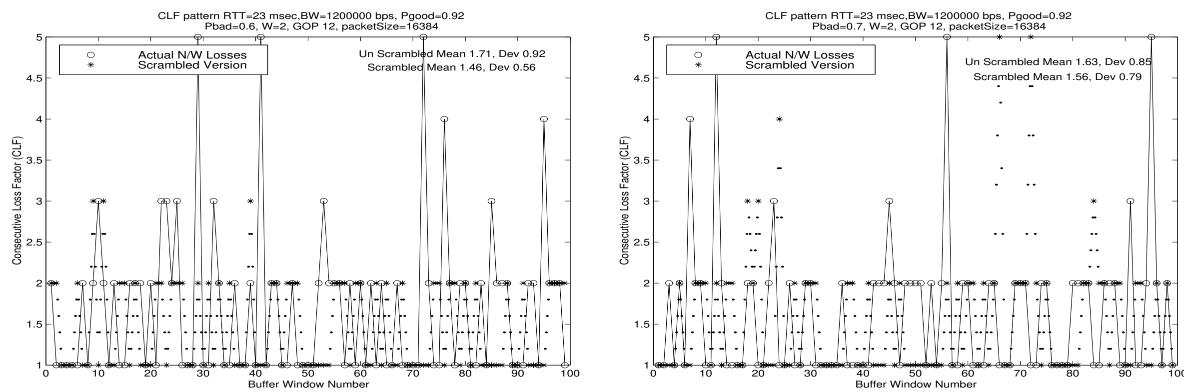


Figure 11: Experiments on Impact of Network Losses

6.3.3 Impact of Available Bandwidth on CLF

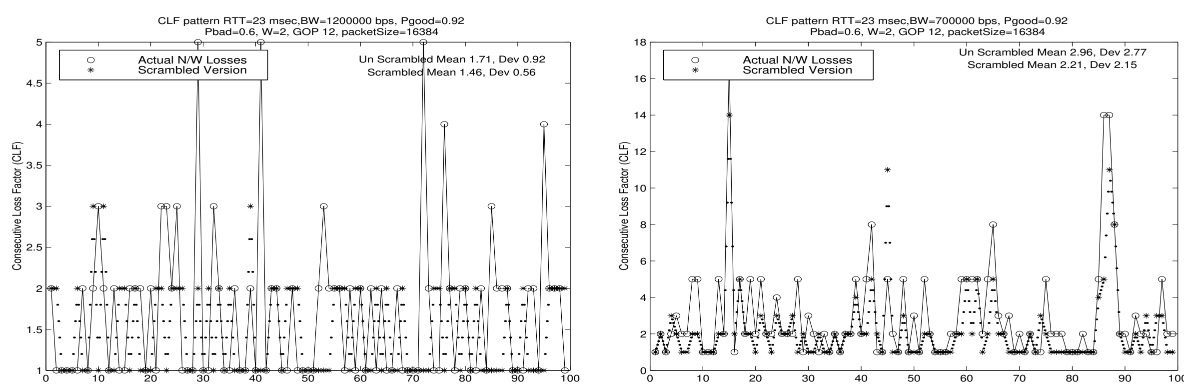


Figure 12: Experiments on Impact of Available Bandwidth

The next two experiments show that our scheme also perform better with different available bandwidth. Here buffer size is kept at 2 GOPs, $Pbad$ is 0.6, and bandwidth is varied from 700 Kbps to 1.2 Mbps. Figure 12 shows results of these experiments. 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 2 which is the threshold for a perceptually acceptable video stream.

6.3.4 Impact of Buffer Size and Startup Delay on CLF

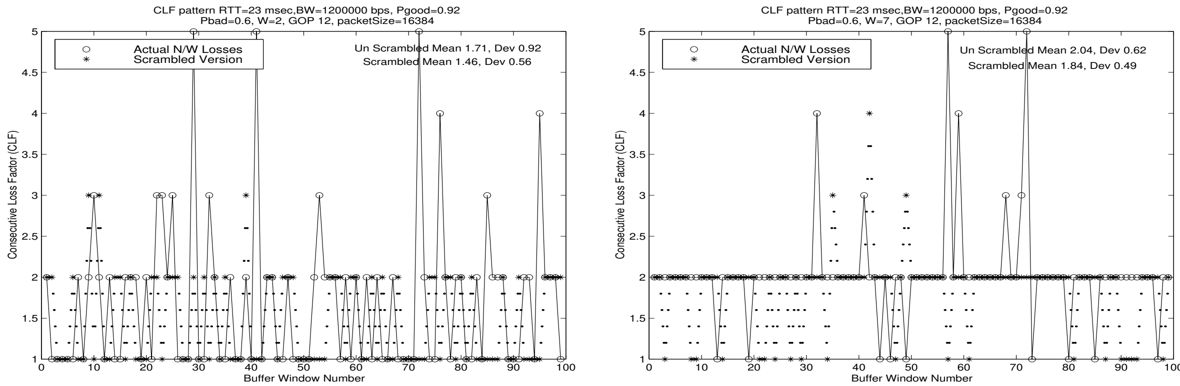


Figure 13: Experiments on Impact of Buffer Size

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 second or 3.5 seconds. Both these values are acceptable in most practical situation. Network loss probability $Pbad = 0.6$ and bandwidth is 1.2 Mbps. Results of these experiment are presented in Figure 13. Again, both mean and deviation of CLF are better. This consistency proves a strong point that error spreading scales well in various scenarios.

6.3.5 When network losses are not bursty

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 come up with a non-bursty network loss pattern, which increases CLFs when permutation scheme is applied.

We must first mention the fact that 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 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. Figure 14 shows two experiments when $Pbad$ is 0.1 and 0.3. Low $Pbad$ is equivalent to low burstiness in network loss. The experiments show that statistically the scrambling scheme is definitely not worse than the non-scrambling counterpart even if the main assumption fails.

7 Conclusions

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 extends 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 non-deterministic network scenario and showed the

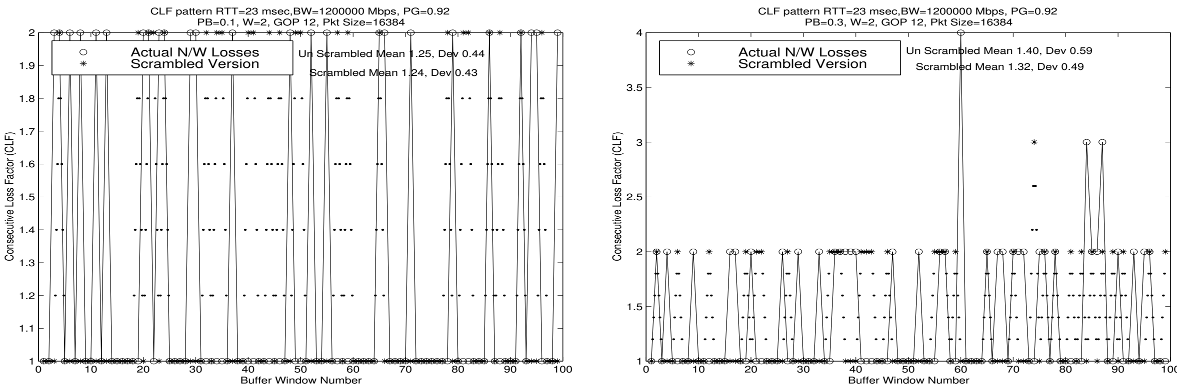


Figure 14: Experiments on Impact of Non-bursty Network Loss

orthogonal nature of our scheme with other CM error handling protocols. Lastly, 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 [7].

References

[1] J. C. Bolot and T. Turletti, "Experience with Control Mechanisms for Packet Video," *ACM Communication Review, Jan '98*, vol. 28, no. 1, 1998.

[2] J. C. Bolot, "End-to-End Packet Delay and Loss Behaviour in the Internet," *ACM SIGCOMM '93, Ithaca, NY*, 1993.

[3] V. Paxson, "End-to-End Internet Packet Dynamics," *ACM SIGCOMM '97, Cannes, France*, 1997.

[4] J. H. Yao, Y. M. Chen, and T. Verma, "MPEG-Based Audio-on-demand experiment for the Internet," *Internetworking '96, Nara, Japan*, 1996.

[5] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, " Modeling TCP Throughput: A Simple Model and its Empirical Evaluation," *ACM SIGCOMM '98, Vancouver, CA*, Sep 1998.

[6] S. Floyd and V. Jacobson, " Random Early Detection gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug 1993.

[7] D. Wijesekera, J. Srivastava, A. Nerode, and M. Foresti, "Experimental Evaluation of Loss Perception in Continuous Media," *to appear in ACM - Springer Verlag*, July 1997.

[8] D. Wijesekera and J. Srivastava, "Quality of Service (QoS) Metrics for Continuous Media," *Multimedia Tools and Applications*, vol. 3, pp. 127–166, September 1996.

[9] D. Wijesekera, S. Varadarajan, S. Parikh, J. Srivastava, and A. Nerode, "Performance evaluation of media losses in the continuous media toolkit," in *International Workshop on Multimedia Software Engineering , MSE'98, Kyoto, Japan*, 1998.

[10] J. C. Bolot, T. Turletti, and I. Wakeman, "Scalable Feedback Control for Multicast Video Distribution in the Internet," *ACM SIGCOMM'94, London, UK*, pp. 58–67, 1994.

[11] S. Ramanathan and P. V. Rangan, "Feedback Techniques for Intra-Media Continuaity and Inter-Media Synchronization in Distributed Multimedia Systems," *The Computer Journal*, vol. 36, no. 1, pp. 19 – 33, 1993.

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

[13] S. McCane, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *ACM SIGCOMM '96*, pp. 117–130, Sep. 1996.

[14] W. C. Geng, F. Jahanian, and S. Sechrest, "An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video," *ACM/Springer-Verlag Multimedia Systems Journal*, 1996.

[15] D. Towsley, "Providing quality of service in packet switched networks," in *Performance Evaluation of Computer Communication Systems* (L. Donatiello and R. Nelson, eds.), pp. 560–586, Springer Verlag, 1993.

[16] R. Aptekar, J. Fisher, V. Kisimov, and H. Nieshlos, "Distributed Multimedia:User Perception and Dynamic QoS," in *SPIE* , vol. 2188, SPIE, 1994.

[17] S. V. Raghavan, S. K. Tripathi, and B. Prabhakaran, "On QoS Parameters and Services for MultiMedia Applications," Tech. Rep. 3167, Department of Computer Science, University of Maryland, College Park, MD 20742 USA, 1994.

[18] B. Wah and D. Lin, "Transformation-based Reconstruction for Audio Transmissions over the Internet," *17th IEEE Symposium on Reliable Distributed Systems, '98*, pp. 211–217, Oct 20-23 1998.

[19] A. Albanese, J. Blomer, J. E. M. Luby, and M. Sudan, "Priority Encoding Transmission," *Proceeding of 35th Annual Symposium on Foundations of Computer Sciences, Santa Fe, New Mexico*, Nov 20-22 1994.

[20] R. Hasimoto-Beltran and A. Khokhar, "Pixel Level Interleaving Schemes for Robust Image Communication," *17th IEEE Symposium on Reliable Distributed Systems, '98*, pp. 455–460, Oct 20-23 1998.

[21] J. C. Bolot and A. V. Garcia, "The case for FEC-based error control for packet audio in the Internet," *To appear in ACM Multimedia Systems*.

[22] J. C. Bolot and Turletti, "Adaptive Error Control for Packet Video in the Internet," *Proceedings IEEE International Conference on Image Processing, ICIP'96, Laussane, Switzerland*, Sep 1996.

[23] I. Rhee and S. R. Joshi, "Error Recovery using FEC and Retransmission for Interactive Video Transmission," Tech. Rep. June, TR-98-12, Dept of Computer Science, North Carolina State University, 1998.

[24] H. Liu and M. E. Zarki, "Performance of video transport over wireless network using hybrid ARQ," in *International Conference on Universal Personal Communications, (ICUPC), Boston, MA*, pp. 567–571, IEEE, October 1996.

[25] W. Luo and M. E. Zarki, "Analysis of error concealment schemes for MPEG-2 video transmission over ATM networks," in *Visual Communications and Image Processing, Taiwan*, SPIE/IEEE, May 1995.

[26] P. Salama, N. B. Shroff, and E. J. Delp, "Error Concealment in Encoded Video," in *submitted to Transactions on Image Processing*, IEEE, 1998.

[27] 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.

[28] H. Schulzrinne, S. Casner, R. Frederick, and S. McCane, "RTP: A Transport Protocol for Real-Time Applications ," *RFC 1889*, 1994.

[29] S. Sedigh, J. Joshi, A. Bashandy, and A. Ghafoor, "Evaluation of Filtering Mechanisms for MPEG Video Communications," *17th IEEE Symposium on Reliable Distributed Systems, '98*, pp. 449–454, Oct 20-23 1998.

[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," *To Appear in IEEE INFOCOM '99*, 1999.

[31] H. Q. Ngo, S. Varadarajan, and J. Srivastava, "On Achieving Lower Consecutive Losses for Continuous Media Streams," Tech. Rep. TR99-005, Dept of Computer Science, University of Minnesota, 1999.

[32] H. Q. Ngo, S. Varadarajan, and J. Srivastava, "Error Spreading: Reducing Bursty Errors in Continuous Media Streaming," in *International Conference on Multimedia Computing and Systems, ICMCS*, IEEE,Multimedia, June 1999.

[33] S. Varadarajan, H. Q. Ngo, and J. Srivastava, "Error Spreading: A Perception-Driven Approach to Error Handling in Continuous Media Streaming," in *submitted to the The seventh ACM Multimedia Conference (Multimedia 99)*, ACM,Multimedia, October 1999.

[34] R. Steinmetz and G. Blakowski, "A media synchronization survey: Reference model, specification and case studies," *IEEE Journal on Selected Areas in Communication*, vol. 14, no. 1, pp. 5–35, 1996.

[35] R. P. Stanley, *Enumerative Combinatorics*, vol. 1. Cambridge, 1997.

[36] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Mc Graw Hill, 1990.

[37] B. Heinrichs and R. Karabek, "Tcp/ip supporting real-time applications: The predictive delay control algorithm," in *Second International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 45–47, 1992.

[38] B. C. Smith, *Implementation Techniques for Continuous Media Systems and Applications*. PhD thesis, University of California, Berkeley, 1994.

[39] B. Smith, "Cyclic-UDP: A Priority Driven Best-Effort Protocol.." http://www.cs.cornell.edu/Info/Faculty/Brian_Smith.html/Publications.

[40] B. Smith, L. Rowe, and S. Yen, "A Tcl/Tk Continuous Media Player," in *Proceedings Tcl 1993 Workshop*, June 1993.

[41] B. Smith, L. A. Rowe, and Y. S., "Tcl distributed programming," in *Proceedings of the 1993 Tcl/Tk Workshop*, 1993.

Appendix: Proof of the Bounded Network Error Case

In this appendix, we shall provide the proofs of two Lemmas concerning two main cases of Theorem 3.2. Section A presents the case where $p \leq \frac{m}{2}$, in which we have more freedom to choose our permutation. Section B discusses the case where $\frac{m}{2} < p < m$.

A Simple case

Lemma A.1. *If* $0 < p \leq \frac{m}{2}$ *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. For example, if $\pi = 4\ 1\ 3\ 6\ 2\ 5$ then $\pi_1 = 4$, $\pi_2 = 1$, but $\pi(1) = 2$ and $\pi(2) = 5$.

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 $p \leq \frac{m}{2}$, we have $p \leq m'$. Moreover, $\gcd(m, m') = 1$, hence $p' \leq m' < \frac{m}{2}$. We now construct a permutation π such that $C^\pi = 1$. Let π be defined as follows.

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

  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 spreaded out in a "better manner". To illustrate this, consider the case where $m = 17$ and $p = 5$. Our choice of p' is 5 in this case, thus in two line notation we have

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 1 & 8 & 15 & 5 & 12 & 2 & 9 & 16 & 6 & 13 & 3 & 10 & 17 & 7 & 14 & 4 & 11 \end{pmatrix}$$

While if we choose $p' = m' = 8$, the permutation becomes

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 1 & 16 & 14 & 12 & 10 & 8 & 6 & 4 & 2 & 17 & 15 & 13 & 11 & 9 & 7 & 5 & 3 \end{pmatrix}$$

Although this permutation does satisfy $C^\pi = 1$, it has the alternative numbers being too close to each other, thus we might end up in a situation where we lose every other frame for a while and then a chunk of successfully arrived frames. This clearly creates worse affect perceptually on the media stream comparing to the previous choice of p'.

- **Case 2 : m is even**

  If $\{j, p \leq j < \frac{m}{2} \wedge \gcd(m, j) = 1\} \neq \emptyset$ 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.

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

  If i is even, then $\pi(i) = \frac{i}{2}$, namely all the even numbers will be placed from the first position to the $\left(\frac{m}{2}\right)^{th}$ position in increasing order. When i is odd, $\pi(i) = p' + \frac{i+1}{2}$, so all the odd numbers will be placed from the $(\frac{m}{2}+1)^{st}$ position to the m^{th} 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

$$\begin{aligned} i &= p'.(\pi_i \bmod 2) + \left\lceil \frac{\pi_i}{2} \right\rceil \\ j &= p'.(\pi_j \bmod 2) + \left\lceil \frac{\pi_j}{2} \right\rceil \end{aligned}$$

  So,

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

  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 $1 \le k \le m-p+1$ 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

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

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 $m - p + 1 < k \le m$, 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, $m - j + i = m - 1 \ge p' \ge p$, contradiction !

$\square$

Example A.2. Let $m = 16$ and $p = 8$, then

$$\pi = \begin{pmatrix} 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \end{pmatrix}$$

B More complicated case

Lemma B.1. *If $\frac{m}{2} < p < m$ then $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$*

Proof. In order to prove that $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$, we will first prove that

$$k_0 \ge \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1 \tag{6}$$

then specify a permutation π so that $C^\pi = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$.

Let $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_m)$ be any permutation on $S = \{1, 2, \ldots m\}$. Let $Y = y_1, y_2, \ldots y_{m-p}$ be the sequence obtained from sorting $\sigma_1, \sigma_2, \ldots \sigma_{m-p}$ in increasing order. Intuitively, Y is the complement of W_{m-p+1}^σ with respect to S.

Now, for convenience let $y_0 = 0$ and $y_{m-p+1} = m + 1$. Let $S_i = \{x : y_{i-1} < x < y_i, x \in S\}$, where $1 \le i \le m - p + 1$. S_i is simply the set of numbers between y_{i-1} and y_i in S.

$$y_0 = 0 \; \underbrace{1\;2\;3\;\ldots}_{S_1} \; y_1 \underbrace{\ldots}_{S_2} \; y_2 \ldots \underbrace{\ldots}_{S_i} \; y_i \;\cdots\; \underbrace{\ldots}_{S_{m-p}} \; y_{m-p} \underbrace{\ldots\; m}_{S_{m-p+1}} \; m+1 = y_{m-p+1}$$

It is clear that

$$\sum_{i=1}^{m-p+1} |S_i| = p$$

Since $S_1 \uplus S_2 \uplus \ldots \uplus S_{m-p+1} = W_{m-p+1}^\sigma$, where $\uplus$ denotes disjoint union, it must be the case that $c_{m-p+1}^\sigma \ge |S_i|,\ \forall i \in \{1, 2 \ldots, m-p+1\}$. Thus, we have

$S - W^{\sigma}_{m-p+1}$	W^{σ}_{m-p+1}	
$\sigma_1, \sigma_2, \dots \sigma_{m-p}$	$\sigma_{m-p+1}, \dots \sigma_p$	$\sigma_{p+1}, \dots \sigma_m$
W^{σ}_1		$S - W^{\sigma}_1$

Table 6: Pictorial illustration of $W^{\sigma}_1, W^{\sigma}_{m-p+1}, S - W^{\sigma}_1$ and $S - W^{\sigma}_{m-p+1}$

$$p = \sum_{i=1}^{m-p+1} |S_i| \leq \sum_{i=1}^{m-p+1} c^{\sigma}_{m-p+1} = (m-p+1).c^{\sigma}_{m-p+1}$$

This implies

$$C^{\sigma} \geq c^{\sigma}_{m-p+1} \geq \left\lceil \frac{p}{m-p+1} \right\rceil$$

this inequality holds for all $\sigma \in S_m$'s, consequently

$$k_0 \geq \left\lceil \frac{p}{m-p+1} \right\rceil \tag{7}$$

Inequality (7) is not as tight as (6). To prove that (6) holds, we need to consider two cases.

Case (i) : $(m-p+1) \not| p$. In this case, $\lceil \frac{p}{m-p+1} \rceil = \lfloor \frac{p}{m-p+1} \rfloor + 1$, so (7) implies (6).

Case (ii) : $(m-p+1) \mid p$. In this case, let $y = \lceil \frac{p}{m-p+1} \rceil = \lfloor \frac{p}{m-p+1} \rfloor$.

Suppose $k_0 = y$ and let $\sigma = \sigma_1, \sigma_2, \dots \sigma_m$ be a particular permutation on S so that $C^{\sigma} = y$. From the definition of C^{σ}, for any $1 \leq i \leq m-p+1$, we must have $c^{\sigma}_i \leq y$. Consider two special sliding windows $W^{\sigma}_1 = \{\sigma_1, \sigma_2, \dots \sigma_p\}$ and $W^{\sigma}_{m-p+1} = \{\sigma_{m-p+1}, \sigma_{m-p+2}, \dots \sigma_m\}$. Table 6 illustrates the situation.

We have $2p > m$, so $p > m-p$. Hence, there is no overlapping between $S - W^{\sigma}_1$ and $S - W^{\sigma}_{m-p+1}$. From the previous analysis, $k_0 = y$ holds if and only if $|S_i| = k_0, \forall i \in \{1, 2, \dots, m-p+1\}$. So it must be the case that

$$y_i = i.(k_0+1), \forall i = 1, 2, \dots m-p \tag{8}$$

because if $\exists i, |S_i| < k_0$, then

$$\begin{array}{rcl} m & = & \sum_{i=1}^{m-p+1} |S_i| + (m-p) \\ & < & (m-p+1).k_0 + (m-p) \\ & = & (m-p+1).\frac{p}{m-p+1} + (m-p) \\ & = & m, \text{ contradiction!} \end{array}$$

The previous analysis was done based on the window W^{σ}_{m-p+1} containing the last p elements of σ. So we have $\{y_1, y_2, \dots \dots y_{m-p}\} = S - W^{\sigma}_{m-p+1}$. Notice that exactly the same result holds if we do the analysis with respect to W^{σ}_1, in which case the y_i's are also determined by (8). So, we have the following :

$$\{1(k_0+1), 2(k_0+1), \ldots, (m-p)(k_0+1)\} = S - W^{\sigma}_{m-p+1} = S - W^{\sigma}_1$$

This is impossible since $(S - W^{\sigma}_1)$ and $(S - W^{\sigma}_{m-p+1})$ are disjoint. Consequently $k_0 \neq y$, so $k \geq y + 1 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$.

Finally, to show that $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$, it is sufficient to specify a permutation π on S such that $C^{\pi} = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$.

Before specifying π, we need to observe some facts. Let $q = m - p$, $r = \left\lfloor \frac{p}{q+1} \right\rfloor$, and $t = \left\lfloor \frac{m}{r+2} \right\rfloor$, then $q \geq 1$ because $q = m - p > 0$. Moreover, since $p > m - p = q$, we have $p \geq q + 1$. Thus $r \geq 1$ and we can write p as :

$$p = \quad (q+1)r + r', \quad 0 \leq r' \leq q$$

In addition, $m = p + q = (q+1)r + r' + q \geq qr + q + r \geq r + 2$, so $t \geq 1$ and we can write m as :

$$m = \quad (r+2)t + t', \quad 0 \leq t' \leq r + 1$$

Now, we specify π in one line notation by considering 2 cases as follows.

Case (i) : $t' = r + 1$.

We have

$$\begin{array}{rrcl} & m & = & p + q \\ \Rightarrow & (r+2).t + (r+1) & = & (q+1).r + r' + q \\ & & = & (q-1).(r+2) + r + r' - q + r + 2 \\ \Rightarrow & (r+2).t + (r+1) & = & (q-1).(r+2) + (r' - q) + 2.r + 2 \\ \Rightarrow & (r+2).t & = & (q-1).(r+2) + (r' - q) + r + 1 \end{array}$$

Moreover, since $r' \leq q$, we get

$$\begin{array}{rrcl} & (r+2).t & \leq & (q-1).(r+2) + r + 1 \\ \Rightarrow & t & \leq & (q-1) + \frac{r+1}{r+2} \end{array}$$

t and q are natural numbers, so

$$t \leq q - 1 \tag{9}$$

We are ready to describe our π now. Notice that we would like C^{π} to be $r+1$, making $k_0 = r+1$ as desired. First we define 2 special arrays $A = \{a_i\}_1^{t+1}$ and $B = \{b_i\}_1^{t+1}$ as follows.

$$\begin{array}{rcll} a_i & = & 1 + (i-1).(r+2), & 1 \leq i \leq t+1 \\ b_i & = & (r+1) + (i-1).(r+2), & 1 \leq i \leq t+1 \end{array}$$

It is obvious that $1 \leq a_i \leq m$ and $1 \leq b_i \leq m$, thus A and B are subsequences of S. Let $C = \{c_i\}_1^{m-2.(t+1)}$ be the increasing sequence of numbers in S but not in A and B. Notice that if we write numbers $1, 2, \ldots m$ in increasing order, we have the following picture :

$$\begin{array}{ccccccccccccccc} & & b_1 & & & & b_2 & & & & & & & & b_{t+1} \\ 1 & .. & r+1 & .. & r+3 & .. & 2r+3 & .. & 2r+5 & .... & 1+t(r+2) & .. & m \\ a_1 & & & & a_2 & & & & a_3 & & a_{t+1} & & \end{array}$$

and the rest of the numbers (the dots in the picture) contribute sequence C. Finally, here is our π in one line notation :

$$\pi = (\underbrace{a_{t+1}, a_t, \ldots a_1}_{A}, \underbrace{c_{m-2.(t+1)}, \ldots\ldots\ldots\ldots, c_1}_{C=S-A-B} \underbrace{b_{t+1}, b_t, \ldots b_1}_{B})$$

To prove that this is a valid permutation, it suffices to observe that the sequences $\{a_i\}$ and $\{b_i\}$ don't intersect, since $a_i \equiv 1 \pmod{r+2}$, while $b_j \equiv r+1 \pmod{r+2}$.

We are left to prove that for every $1 \leq k \leq m$, W^π doesn't contain more than $r+1$ *consecutive* integers, where as usual, when $m-p+1 < k \leq m$ we interpret *consecutiveness* a little differently.

Notice that $\forall i \in \{1, \ldots, t\}$, the numbers of integers between a_i and a_{i+1} exclusively is $r+1$, which we shall call the *internal distance* between a_i and a_{i+1}. Similarly, the internal distance between b_i and b_{i+1} is also $r+1$. Now, consider the first sliding window W_1^π of size p of π :

$$\pi = \overbrace{\underbrace{a_{t+1}, a_t, \ldots a_1, c_{m-2.(t+1)}, \ldots\ldots,}_{W_1^\pi \longrightarrow} \ldots, c_1}^{m-(t+1)\ numbers}, b_{t+1}, b_t, \ldots b_1$$

This window can not contain any b_i's, since from (9) we have $|W_1^\pi| = p = m - q \leq m - (t+1)$. As we have noticed earlier, the internal distance between b_i and b_{i+1} is $r+1$ for all $i \in \{1, 2, \ldots t\}$, thus it is easy to see that $c_1^\pi \leq r+1$. We will prove that this property holds for all sliding windows W_k^π by induction on k, where $1 \leq k \leq m-p+1$, namely $\forall k \in \{1, 2, \ldots m-p+1\}, c_k^\pi \leq r+1$.

- Base case : $c_1^\pi \leq r+1$ as discussed.
- Suppose we have $c_k^\pi \leq r+1$ for some $1 \leq k < m-p+1$.
- Let's look at W_{k+1}^π. If W_{k+1}^π does not contain any element of B then $c_{k+1}^\pi \leq r+1$ holds trivially. Otherwise, suppose W_{k+1}^π does contain some set of more than $r+1$ consecutive integers. Let i be the least integer such that $b_i \in W_{k+1}^\pi$. It is clear that W_k^π doesn't contains b_i, because intuitively we have just moved W_k^π one step to the right of π and added b_i into W_k^π to obtain W_{k+1}^π. The only way for W_{k+1}^π to contain a set of more than $r+1$ consecutive integers is when this set contains b_i, since by induction hypothesis, just before this point, $c_k^\pi \leq r+1$. Additionally, since the size of W_{k+1}^π is p, it's easy to see that if $b_i \in W_{k+1}^\pi$ then $a_j \notin W_{k+1}^\pi, \forall j \geq i$.

  However, if b_i is contained in some set X of more than $r+1$ consecutive integers, then X has to either contain a_i or a_{i+1} because $a_i < b_i < a_{i+1}$ and the internal distance between a_i and a_{i+1} is $r+1$. Contradiction !

  So W_{k+1}^π does not contain any set of more than $r+1$ consecutive integers. In other words, $c_{k+1}^\pi \leq r+1$.

We also have to show that

$$c_k^\pi \leq r+1, \text{ when } m-p+1<k\leq m \tag{10}$$

- If $r=1$, then $m=3t+2$; thus, $2(t+1)=m-t\geq p+1$. Consequently, none of $W_k^\pi, m-p+1< k\leq m$ contains both $a_1=1$ and $b_{t+1}=m$. (10) follows trivially.
- If $r>1$, then $c_1=2$ and $c_{m-2(t+1)}=m-1$. Moreover, clearly $c_{m-2(t+1)}\notin\{\pi_k,\dots\pi_m\}$ and $c_1\notin\{\pi_1,\dots\pi_{p+k-m-1}\}$ whenever $m-p+1<k\leq m$. Thus, for these values of k, c_k^π is at most 2, which is at most $r+1$, which completes our proof.

Example B.2. Let $m=17$, $p=9$. Thus, $k_0=\left\lfloor\frac{p}{m-p+1}\right\rfloor+1=\left\lfloor\frac{9}{17-9+1}\right\rfloor+1=2$. $r=1$ and $17=3.5+2$, so $t'=2=r+1$ in this case. The sequences A and B are as follows. $A=1,4,7,10,13,16$. $B=2,5,8,11,14,17$. And lastly, applying our scheme gives us permutation

$$\pi=\begin{pmatrix} 16 & 13 & 10 & 7 & 4 & 1 & 15 & 12 & 9 & 6 & 3 & 17 & 14 & 11 & 8 & 5 & 2 \end{pmatrix}$$

Case (ii) : $0\leq t'<r+1$.

We have

$$\begin{aligned} m &= p+q \\ \Rightarrow\quad (r+2).t+t' &= (q+1).r+r'+q \\ &= q.(r+2)+r+r'-q \\ \Rightarrow\quad (r+2).t &= q.(r+2)+(r'-q)+r-t' \end{aligned}$$

Moreover, since $r'\leq q$, we get

$$\begin{aligned} (r+2).t &\leq q(r+2)+r-t' \\ \Rightarrow\quad t &\leq q+\tfrac{r-t'}{r+2} \end{aligned}$$

t and q are natural numbers, so

$$t\leq q \tag{11}$$

Similar to the case (i), we define 2 special arrays $A=\{a_i\}_1^t$ and $B=\{b_i\}_1^t$ as follows.

$$\begin{aligned} a_i &= t'+1+(i-1).(r+2) && 1\leq i\leq t \\ b_i &= i.(r+2) && 1\leq i\leq t \end{aligned}$$

It is obvious that $1\leq a_i\leq m$ and $1\leq b_i\leq m$, thus A and B are subsequences of S. $A\cap B=\emptyset$ since $(r+2)$ divides a_i and does not divide b_j for all i and j. Let $C=\{c_i\}_1^{m-2t}$ be the increasing sequence of numbers in S but not in A and B. If we write numbers $1,2,\dots m$ in increasing order, this time we have the following picture :

$$\begin{array}{ccccccccccc} & & & & b_1 & & & & b_t & & \\ 1 & .. & t'+1 & .. & r+2 & .... & t'+1+(t-1)(r+2) & .. & t(r+2) & .. & m \\ & & a_1 & & & & a_t & & & & \end{array}$$

and the rest of the numbers (the dots in the picture) contribute sequence C. π in one line notation is as follows.

$$\pi = (\underbrace{a_t, a_{t-1}, \ldots a_1}_{A}, \underbrace{c_{m-2t}, \ldots\ldots\ldots\ldots, c_1}_{C=S-A-B} \underbrace{b_t, b_{t-1}, \ldots b_1}_{B})$$

The proof goes almost the same as in case (i). Due to $t \leq q$, W_1^π does not contain any of the b_i's. Straightforward induction shows that when $1 \leq k \leq m-p+1$, none of W_k^π contain more than $r+1$ consecutive integers. On the other hand, when $m-p+1 < k \leq m$, we always have

$$c_1 \notin \{\pi_1, \ldots, \pi_{p+k-m-1}\} \text{ and } c_{m-2t} \notin \{\pi_k, \ldots, \pi_m\}$$

If $t' = 0$, then $c_1 = 2$ and $c_{m-2t} = m-1$, so $c_k^\pi \leq 2 \leq r+1$. If $t' > 0$, then $c_1 = 1$ and $c_{m-2t} = m$, so $c_k^\pi = 0$.

Example B.3. Let $m = 17$, $p = 12$. $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1 = \left\lfloor \frac{12}{17-12+1} \right\rfloor + 1 = 3$. $r = 2$ and $17 = 4.4 + 1$, so $t' = 1 < r+1$ in this case. The sequences A and B are as follows. $A = 4, 8, 12, 16$. $B = 2, 6, 10, 14$. Applying our scheme gives us permutation

$$\pi = \left(\begin{array}{ccccccccccccccccc} 16 & 12 & 8 & 4 & 17 & 15 & 13 & 11 & 9 & 7 & 5 & 3 & 1 & 14 & 10 & 6 & 2 \end{array}\right)$$

$\square$