An Adaptive, Perception-Driven Error Spreading Scheme 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

For transmission of continuous media (CM) streams such as audio and video over the Internet, a critical issue is that periodic network overloads cause bursty packet losses. Studies on human perception of audio and video streams have shown that bursty losses have the most annoying affects. Hence, addressing this issue is critical for multimedia applications such as Internet phone, video conferencing, distance learning, etc. Classical error handling schemes like retransmission and forward error recovery have the undesirable effects of (a) introducing timing variations, which is unacceptable for isochronous traffic, and (b) using up valuable bandwidth, potentially exacerbating the problem. This paper introduces a new concept called error spreading*, which is a transformation technique that takes an input sequence of packets (from an audio or video stream) and permutes them before transmission. The packets are then un-permuted at the receiver before delivery to the application. The purpose is to spread out bursty network errors, in order to achieve better perceptual quality of the transmitted stream. Analysis has been done to determine the provable lower bound on bursty errors tolerable by this class of protocols. An algorithm to generate the optimal permutation for a given network loss rate is presented. While our previous work had focused on streams with no inter-frame dependencies, e.g. MJPEG encoded video, in this paper the technique is generalized to streams with inter-frame dependencies, e.g. MPEG encoded video. It is also shown how the error spreading approach can be used as a module to be* plugged *into any existing CM transmission protocol to help improve the perceptual quality of the data stream without changing the bandwidth requirement of the base protocol. Thus, this approach is orthogonal to traditional error handling techniques. Finally, results from an experimental evaluation are presented to validate our claims.*

Keywords: Quality of service, Multimedia, Error Spreading, Network bursty error, Adaptive, Permutation scheme

1 Introduction

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], leading to significant bursty errors [3].This bursty loss behavior has been shown to arise from the drop-tail queuing discipline adopted in many Internet routers [4]. 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 [5]. Nevertheless since drop-tail queuing discipline is still adopted in many routers [4], 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 [6]. 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 . 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, 7]. The feedback control can be based on stream rate [8, 9], bandwidth [10], loss/delay [1] and a wide variety of network QoS parameters [11]. Client initiated reaction occurs in reconstruction based schemes like [12]. 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 ([13] and *Forward Error Correction Codes* [14]). There exist hybrid schemes which use both forms of redundancy [15]. Another technique that is called *error concealment* [16], 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 [17, 18]. Services like RTP/RSVP require that some resource allocation and/or reservation mechanism be provided by the network [1].

Our earlier work [19, 20] has proposed 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 continuous media viewing [6] 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 [3]. However, the impact of packet scrambling on user perception or system resources was not investigated. In [19, 20] 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 [19, 20] works only for streams with no inter-frame dependency, e.g. MJPEG video, audio etc. In this paper 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.

In this paper we make several contributions. First, 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 in [19, 20], to include streams with inter-frame dependency. Second, we provide a general error spreading technique which can handle these types of streams. Third, we provide the error spreading protocol which can be used orthogonal to other error handling protocols. We also provide details of the implementation of our protocol in the *Berkeley Continuous Media Toolkit (CMT)*. Finally, we present results of an experimental evaluation that illustrates the benefits of the proposed scheme.

This paper is organized as follows: Section 2 discusses background knowledge and existing results related to our problem. Section 3 presents the general solution to streams with inter-frame dependencies. Section 4 provides a protocol which can be plugged into other error handling schemes. Section 5 presents results of simulations which validate our claim. Finally, section 6 concludes the paper.

2 Background

This section briefly discusses the content based continuity QoS metrics introduced in [21]. Then, we give motivations for our work and existing results on the problem.

2.1 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 [22]). 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]. Given a rate for streams consisting of these LDUs, we envision that there is time slot for each LDU to be played out. In the ideal case a LDU should appear at the beginning of its time slot. In this paper, we use only the content based continuity metrics proposed in [21], and issues arising out of rates and drifts (discussed in [21]) are not considered. Note also that we shall use the term LDU and frame interchangeably.

Figure 1 is from [21]. 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 [21].) 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 [21].

In a user study [6] 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 Motivations

One of the most important QoS parameters that affects the quality of a CM stream is the *Consecutive Loss Factor* [21] (CLF). Packet transmission on networks shows a bursty behavior, alternating between spurts of packet loss and packet delivery [3, 2, 4]. 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 if we permute this sequence before transmission so that consecutive frames become far apart, then 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 boxed numbers represent lost frames.

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

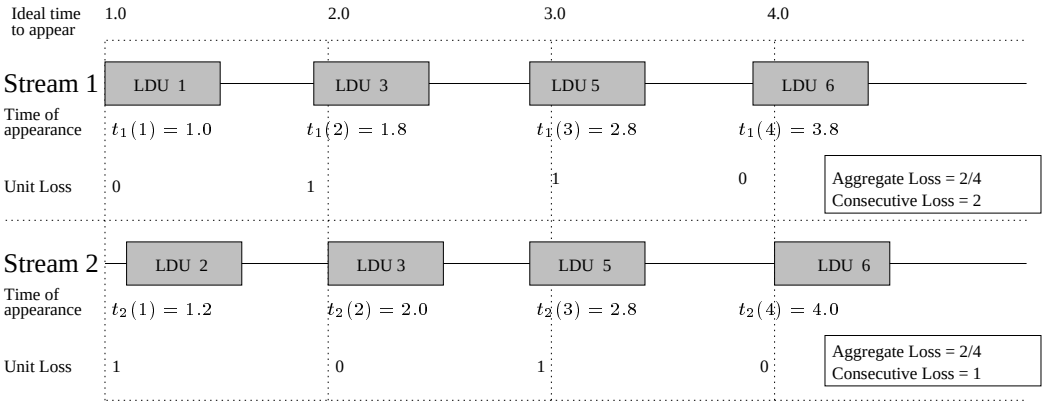

Figure 1: Two Example Streams used to Explain Metrics

	Frame sequence	Consecutive Loss/ Window Size
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 | 02 | 03 04 | 05 | 06 | 07 | 08 09 | 10 | 11 | 12 | 13 | 14 | 15 16 | 17	$1/17$

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

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 [21].
- 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) among 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 hence any permutation scheme must address this issue.

In this paper we address the issues listed above.

2.3 Existing Results

In [20, 19], we have solved this problem, when (a) there is no inter-frame dependency and (b) if the network loss is known (deterministic). We developed a protocol which uses the error spreading idea for continuous media (with no dependency) streaming, using the results of the deterministic case. Due to lack of space, we do not provide the proof of Theorem 1 here. Details of the proof can be found in [23, 19]. For completeness, we briefly summarize our earlier work as follows.

Problem: Bursty Error Reduction Problem (BERP)

- **Objective :** To reduce bursty error, i.e. CLF, to perceptually acceptable level (by spreading it out over the stream) in continuous media streaming.
- **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.
- **Output :** a permutation function f on $S = \{1, 2, 3, \ldots, m\}$ which decides the order in which a set of m consecutive LDUs should 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.

Solution: The following theorem, whose proof is given in [20], summarizes our results. We call our scrambling scheme *k-Cyclic Permutation Order* or *k-CPO* for short, where k is the user's maximum acceptable CLF.

Theorem 1 *Given m is sender's buffer size (fixed) and p is the bound on network bursty loss, 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* $0 < p < m$.

Algorithm $calculatePermutation(m, p)$ (see [20]) gives the appropriate permutation function, based on k-CPO, for a sender's buffer of size m and a bound on network bursty loss p. This result sets up the background for us to solve the more general problem when there are inter-frame dependencies.

3 Modeling Streams with Inter-Frame Dependency

This section is organized as follows. Section 3.1 defines related concepts from Combinatorics, 3.2 discusses our solution for MPEG streams, and section 3.3 generalizes the analysis for any stream with inter-frame dependencies.

3.1 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 [24].

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 [25]. Thus, a linear extension of a poset can be obtained by a topological sort algorithm.

3.2 Analysis of the MPEG case

Figure 2 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 2, the GOPs have size 12. The dashed 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 the 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.

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 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 2 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 currently being decoded.

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. 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

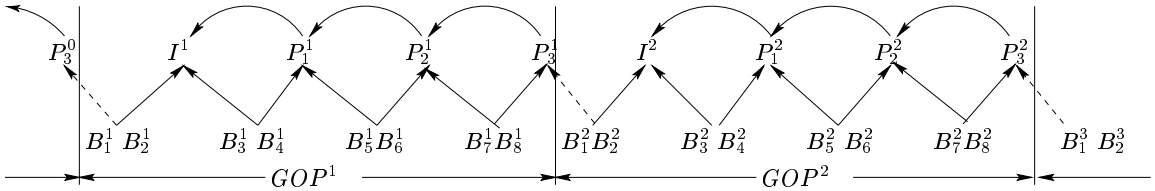

Figure 2: A Sample Pattern of MPEG Frame Dependency

by figure 3. Here, f is the permutation function generated by the $calculatePermutation$ algorithm.

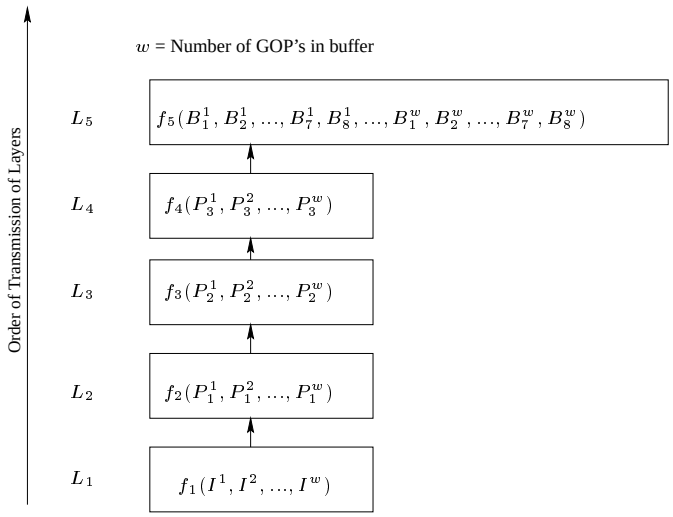

Figure 3: Layered Permutation Transmission Order for MPEG

3.3 General 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.

4 Transmission Protocol

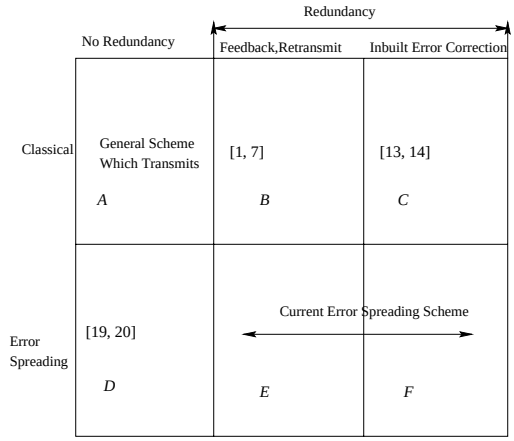

Figure 4: Classification of Error Handling Schemes

Perceptual studies have established the tolerance levels for discontinuity in CM viewing [6, 21], beyond which user dissatisfaction increases 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 4. Our earlier work [20, 19] 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 2, 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 4) 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 [15]. In Section 3 we provided a scrambling order for dependent streams like MPEG, which improves CLF. The goal of this section is to show that error spreading technique 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 4.1 addresses the buffer requirement for the use of our protocol. Section 4.2 discusses how our protocol functions. Lastly, section 4.3 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.

4.1 Buffer Requirements

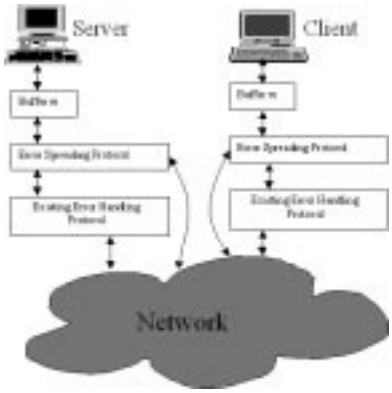

Figure 5: Protocol Setting

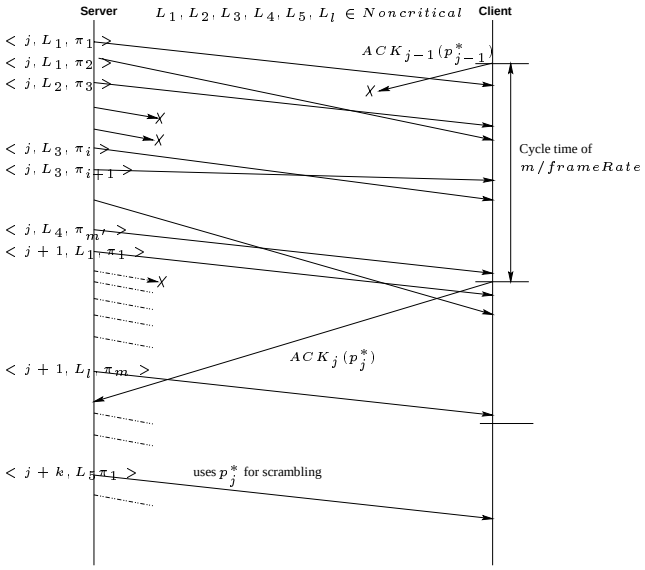

Figure 6: a Protocol Illustration

As shown in Figure 5, the server and client both require a buffer of size $m = w * F_{GOP} * MFS$, where F_{GOP} = Number of frames in a GOP, MFS = Maximum Frame Size and w = Number of GOPs, $w \geq 2$. To illustrate, we took the MPEG traces from `ftp://gaia.cs.umass.edu/pub/zhzhang/`. The maximum GOP sizes in bits are: *Jurassic Park*-62776, *Silence of the Lambs*-462056, *Star Wars*-932710, *Terminator*-407512, and *Beauty and the Beast*-769376. 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, $m = 2 * F_{GOP} * MFS = 2 * MaxGOPSize = 2 * 113 = 226 Kbytes$ is quite viable.

4.2 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 3, $l = 5$. 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 3, $Critical = \{L_1, L_2, L_3, L_4\}$ 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 $calculatePermutation(|L_i|, p)$; details of which are given in the appendix. $|L_i|$ 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 [26]. However it has been shown in [27] 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 [28, 29], 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

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

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 5.

Figure 6 illustrates an example of how client and server interact during transmission of layer $L_i \in Noncritical$. 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, 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 [20]. Also as shown earlier, our approach works for any encoding scheme, even though the illustrations are with MPEG.

4.3 Error Spreading as an Orthogonal Dimension

As seen from the earlier section 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 5). It is clear that any error handling protocol using retransmission or forward error correcting codes, e.g. [1, 7, 11, 30, 14, 15], or error concealment protocol [16], only decreases the $Critical$ set and increases the $Noncritical$ set. Thus, being aware of the type of reconstruction of frames possible on the client side as well as the protocol mechanism for transmission, only fixes q (as defined in Section 4.2). 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 .
- 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.

4.4 Implementation on CMT

To investigate the feasibility of implementing the above features, the error spreading scheme was implemented on the *Berkeley Continuous Media Toolkit (CMT)* [29]. Summary of CMT's relevant architechture can be found in [23]

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) picks up frames from the common buffer, decides which frames in the buffer are 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 being developed by Daishi Harada). An example is shown in table 2. Note that frame dropping can potentially occur at any of the previously mentioned objects, i.e. if the object decides it cannot transmit the given frame using the estimated available resources, or if a frame playback deadline has elapsed.

We replaced IBO with our error spreading algorithm (based on k-CPO) presented earlier. Since k-CPO has been proven to be optimal, it is better than IBO in all cases. Table 2 shows an example. Also, CMT provides a handle onto the buffer size, parameter W, by allowing the user to vary the cycle time of the LTS objects [23]. Losses of B frames occur only in the tail of the set of B frames because of the way the CMT protocol works. It sends a bunch of frames at the head

	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	$3/8$
k-**CPO**	01 04 07	02 05 08 03 06	$2/8$

Table 2: 8 frames ordering of IBO and one of the cases of our scrambled order

of the buffer (which has I,P and B frames in that order). The number of frames depends on its estimated parameters. It is easily seen that as long as the number of frames lost due to network losses is less than half the number of B frames sent, IBO provides good CLF. As can be seen in table 2, in a pathological network scenario wherein the number of frames lost is greater than half the number of B frames sent, IBO performance starts degrading, while in k-CPO still provides the good CLF (we adhere to the bounds specified in theorem 1). Further in our layered order, we also use our ordering mechanism among the I and P frames to ensure 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 specific encoding. We have provided the details in section 3. 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.

5 Experimental Evaluation

In this section, we present the results of a detailed simulation based evaluation of our protocol using the layered permutation transmission order for MPEG described in section 3.2, and compare it against the usual MPEG transmission model.

5.1 Simulation Model

Network loss pattern is modeled by a two state Markov model as shown in Figure 7

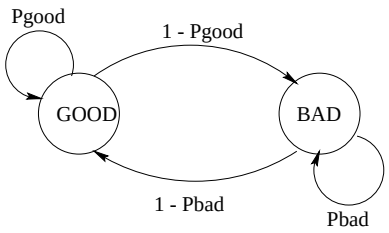

Figure 7: Two State Markov Model of Network Loss Pattern

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 $m = w * F_{GOP}$ 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://ftp.gaia.umass.edu/ . 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 variation is the network packet losses modeled with random derivate 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.

5.2 Simulation Results

Due to lack of space, we show here only the resulting figures with respect to network loss. Other figures can be found in [23].

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 8 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 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 11 in [23] 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.

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 is presented in Figure 12 of [23]. Again, both mean and deviation of CLF are better. This consistency proves a strong point that error spreading scales well in various scenarios.

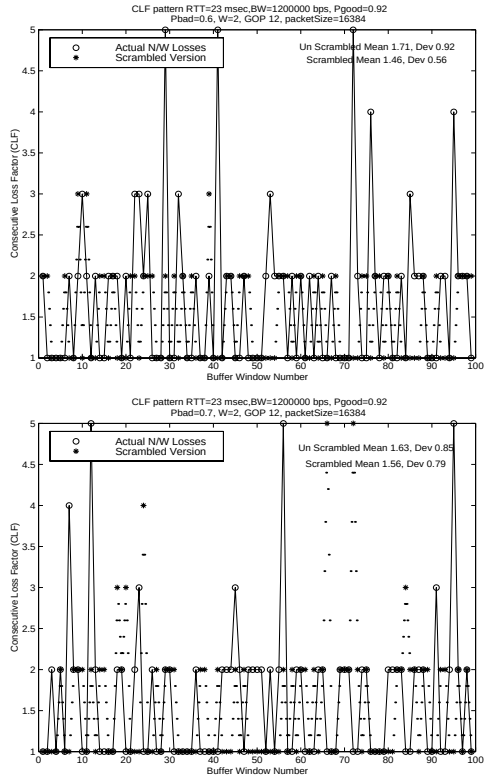

Figure 8: Experiments on Impact of Network Losses

6 Concluding Remarks

In this paper, we have introduced a new concept called *error spreading*, which is a transformation technique that takes an input sequence of packets (from an audio or video stream) and permutes them before transmission. The purpose is to spread out bursty network errors, in order to achieve better perceptual quality of the transmitted stream. We have formulated the problem and gave a general solution for CM streams with inter-frame dependencies. It was also shown how the error spreading approach can be used as a module to be *plugged* into any existing CM transmission protocol to help improve the perceptual quality of the data stream without changing the bandwidth requirement of the base protocol. This approach is thus orthogonal to traditional error handling techniques. Lastly, several experiments were conducted and presented to validate our claims.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

[22] 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.

[23] S. Varadarajan, H. Q. Ngo, and J. Srivastava, "Error Spreading: A Perception Driven Approach Orthogonal to Error Handling in Continuous Media Streaming," Tech. Rep. TR99-028, Dept of Computer Science, University of Minnesota, 1999.

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

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

[26] 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.

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

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

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

[30] 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*.