Error Spreading: Reducing Bursty Errors in Continuous Media Streaming

Hung Q. Ngo, Srivatsan Varadarajan, and Jaideep Srivastava
Department of Computer Science, University of Minnesota,
200 Union street, EE/CS Building, room 4-192, Minneapolis, MN 55455
e-mail: {hngo, varadara, 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. Perceptual studies have shown that users are much more tolerant of a uniformly distributed loss of low magnitude. We next describe a continuous media transmission protocol based on this idea, and validate its performance through an experiment performed on the Internet.

Keywords: multimedia, network bursty error, scrambling 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].

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, 7, 8]. 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 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, 9, 10]. The feedback control can be based on stream rate [11, 12], bandwidth [13], loss/delay [1] and a wide variety of network QoS parameters [14, 15, 16]. Client initiated reaction occurs in reconstruction based schemes like [17, 18]. 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 ([19] and *Forward Error Correction Codes* [20, 21]). Each of these classes of schemes requires extra bandwidth, for feedback and retransmission in the first, and for extra bits in the second category. This need for extra bandwidth can exacerbate the problem, especially since network congestion is the principal culprit for the bursty errors. One more approach that has been proposed, is to provide the real time needs of CM applications over other services like *RSVP and RTP*, which offer varying degree of performance guarantees for applications [22, 23]. Services like RTP/RSVP require that some resource allocation and/or reservation mechanism be provided by the network [1]. Since these mechanisms are not yet widely deployed in the Internet, our focus has not been on them.

Recent work ([24, 25]) has proposed schemes where the overall characteristics of the data being transmitted can be used to control the transmission error. For e.g., [25] 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 [26]. Since drop-tail queuing discipline is still adopted in many routers [5], this scheme may not be directly applicable yet.

In this paper we propose a new type of scheme for handling bursty errors, which we call *error spreading*. A key advantage of this scheme is that it is not based on redundancy, and hence requires absolutely no extra bandwidth. The main idea is that we do not try to reduce overall error, but rather tradeoff bursty error (the *bad error*) for average error (the *good error*). Perceptual study of continuous media viewing [6, 7, 8] 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 has been taken by [4]. They have used a random scrambling technique with redundant reconstruction for audio, and as stated by them they have not investigated the buffer requirements. We have established clearly the bounds and the relationship between buffer requirement and user perceived quality in a bounded bursty network error scenario.

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. Next, we use this analysis to develop such a protocol for networks where there is no bound on the error. 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 transmission protocol, while Section 5 describes its evaluation. Section 6 concludes the paper.

2 Problem Formulation

This section briefly discusses the content based continuity QoS metrics introduced in [7]. Then, we define our problem based on the metrics introduced.

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 [27]). 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 [7], and issues arising out of rates and drifts (discussed in [7]) are not considered. Note also that we shall use the term LDU and frame interchangeably.

The above figure is from [7]. 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 [7].) 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 [7].

In a user study [6] it has been determined that the tolerable value for consecutive losses is two frames. For audio this limit was about three frames.

2.2 Problem Statement

One of the most important factors that affect the quality of a CM stream is the *Consecutive Loss Factor* [7] (CLF). Networks often lose frames in bursts, alternating between lossy burst and successful burst [4, 2, 3, 5]. This often causes unacceptably high CLF. Our objective is to decrease the CLF given the same network characteristics. The main idea is *loss spreading*, or distributing consecutive loss over some time period. The idea is best illustrated by the example in Table 1. The network channel loses 5 consecutive frames numbered from 6 to 10. Thus, sending 12 frames in sequence causes CLF of 5, while sending them in a scrambled sequence causes a CLF of only 1. We now formally state the problem.

Bursty Error Reduction Problem (BERD)

- **Objective :** to reduce the bursty error, i.e. CLF, to a perceptually acceptable level (by spreading it out over the stream).
- **Input parameters:**
  - m is the sender's buffer size, in terms of LDUs. m is determined by the sender's operating environment and its current status.

[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 second.

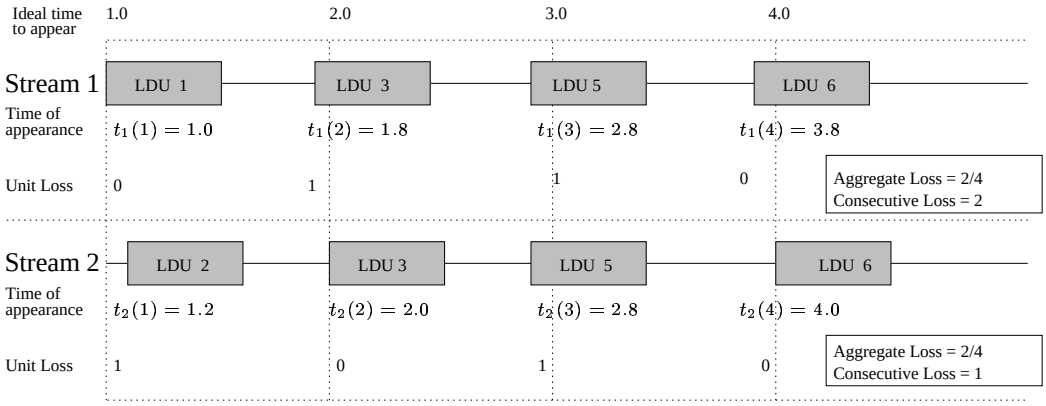

Figure 1: Two Example Streams used to Explain Metrics

	Frame sequence sent			Consecutive Loss / Window Size
In Sequence	01 02 03 04 05	06 07 08 09 10	11 12	$5/12$
Scrambled	01 06 11 04 09	02 07 12 05 10	03 08	$1/12$

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

  - p is the upper bound on the size of a bursty loss in the communication channel, within a window of m LDUs (we relax this assumption in section 4).
  - 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.

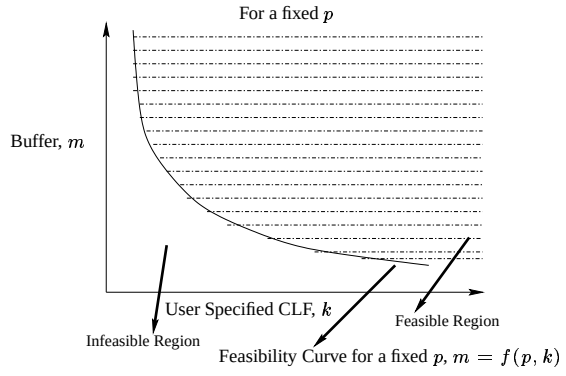

Figure 2: Part of Deterministic Solution Space

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 it 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 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 Bounded Network Error Case

In this section, we will discuss the cases where continuous network loss p is bounded. The following theorem summarize our work on the deterministic cases.

Theorem 1 *If p is bounded and m is fixed, then*

- $k_0 = 0$ *when* $p = 0$
- $k_0 = m$ *when* $p \geq m$
- $k_0 = 1$ *when* $0 < p \leq \frac{m}{2}$.

- $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$ *when* $0 < \frac{m}{2} < p < m$.

Proof: The first two cases are trivially satisfied. The other two cases are immediate from the lemmas and theorems (which we are not providing because of lack of space) given in [28]. Note that if the desired k_0 is given, these formulas allow us to also find the minimum buffer size m_0 to achieve k_0.

Algorithm $calculatePermutation(m, p)$ specifies the appropriate permutation function for the four cases considered in theorem 1. (Note: % denotes mod)

calculatePermutation(m, p)
if $p \leq 0$ or $p \geq m$ **then**
do nothing (the bounds are assumed to be known).
end if
if $p \leq \left\lfloor \frac{m}{2} \right\rfloor$ **then**
if m is odd **then**
for $i \leftarrow 1$ to m **do**
$f(i) \leftarrow ((i-1).p' \,\%\, m)+1$
end for
else
for $i \leftarrow 1$ to m **do**
$f(i) \leftarrow p'.(i\%2) + \left\lceil \frac{i}{2} \right\rceil$
end for
end if
else
$q \leftarrow m - p$
$r \leftarrow \left\lfloor \frac{p}{q+1} \right\rfloor$
$t \leftarrow \left\lfloor \frac{m}{r+2} \right\rfloor$
$t' \leftarrow m\%(r+2)$
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
$C \leftarrow \{1, ..m\} - \{a_i\} - \{b_i\}$
C is extracted in increasing order.
for $i \leftarrow 1$ to $t + 1$ **do**
$f(a_{t+2-i}) \leftarrow i$
end for
for $i \leftarrow t + 2$ to $m - (t + 1)$ **do**
$f(c_{m-i-t}) \leftarrow i$
end for
for $i \leftarrow m - t$ to m **do**
$f(b_{m-i+1}) \leftarrow i$
end for
else
for $i \leftarrow 1$ to t **do**
$a_i \leftarrow i.(r+2)$
$b_i \leftarrow t'+1+(i-1).(r+2)$
end for
$C \leftarrow \{1, ..m\} - \{a_i\} - \{b_i\}$
C is extracted in increasing order.
for $i \leftarrow 1$ to t **do**
$f(a_{t+1-i}) \leftarrow i$
end for
for $i \leftarrow t + 1$ to $m - t$ **do**
$f(c_{m-i-t+1}) \leftarrow i$
end for
for $i \leftarrow m - t + 1$ to m **do**
$f(b_{m-i+1}) \leftarrow i$
end for
end if
end if

Notice that the algorithm take only linear time. As can be seen from the algorithm, the proposed permutation functions are divided into various cases depending on the relationships between m and p. Here we lists several examples to illustrate these cases.

- **Example 1:** $0 < p \leq \frac{m}{2}$ and m is odd. Let $m = 17$ and $p = 8$. Applying our permutation on $S = \{1, 2, \ldots 17\}$ gives us the following sequence :
  $S' = (1\ 6\ 11\ 16\ 4\ 9\ 14\ 2\ 7\ 12\ 17\ 5\ 10\ 15\ 3\ 8\ 13)$

- **Example 2:** $0 < p \leq \frac{m}{2}$ and m is even. Let $m = 16$ and $p = 8$. Applying our permutation on $S = \{1, 2, \ldots 16\}$ gives us the following sequence.
  $S' = (2\ 4\ 6\ 8\ 10\ 12\ 14\ 16\ 1\ 3\ 5\ 7\ 9\ 11\ 13\ 15)$
  As can be seen, there are no two consecutive integers in any sliding window of size 8 or less ($k_0 = 1$ in both cases).

- **Example 3:** $0 < \frac{m}{2} < p < m$, and $m \bmod (\left\lfloor \frac{p}{m-p+1} \right\rfloor + 2) = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$. Let $m = 17, p = 9$. Thus, $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1 = \left\lfloor \frac{9}{17-9+1} \right\rfloor + 1 = 2$. The resulting permutation is
  $S' = (16\ 13\ 10\ 7\ 4\ 1\ 15\ 12\ 9\ 6\ 3\ 17\ 14\ 11\ 8\ 5\ 2)$
  Thus, no sliding window of size 9 contains more than $k_0 = 2$ consecutive integers.

- **Example 4:** $0 < \frac{m}{2} < p < m$, and $m \bmod (\left\lfloor \frac{p}{m-p+1} \right\rfloor + 2) < \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1$. Let $m = 17, p = 12$, then $k_0 = \left\lfloor \frac{p}{m-p+1} \right\rfloor + 1 = \left\lfloor \frac{12}{17-12+1} \right\rfloor + 1 = 3$. Applying our scheme gives us permutation :
  $S' = (16\ 12\ 8\ 4\ 17\ 15\ 13\ 11\ 9\ 7\ 5\ 3\ 1\ 14\ 10\ 6\ 2)$
  and no sliding window of size 12 contains more than $k_0 = 3$ consecutive integers.

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 Unbounded Network Error Case

4.1 Permutation adjustment protocol

Our protocol is a simple feedback based protocol. Some CM systems use TCP/IP for communication [29]. But it has been shown in [30] that CM applications based on TCP are unstable when the real time bandwidth requirements fall below available bandwidth. Thus in this protocol, we use the UDP communication model (like [31, 32]). We dynamically use the solution provided in the deterministic cases as a mechanism for the non-deterministic scenario presented here. We assume that m, the buffer size, is known in advance by both client and server. This can also be part of 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, then initiates transmission of the frames in the buffer. Server changes the permutation scheme based on client's feedback. The permutation scheme changes only at the start of the next buffer of frames.

At the client side, the client waits for a period of $Cycletime = m/frameRate$ (time needed for the client's buffer to be filled up) and calculates consecutive network loss for this buffer window. The client keeps track of the previous window's estimated network consecutive loss and sends its next estimation back 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 decision based on the maximum sequence numbered ACK.

Given a buffer of size m, initially the server assumes the average case where $p = \left\lfloor \frac{m}{2} \right\rfloor$. Denote p_i as the actual consecutive network loss, and p_i^* as the estimated network loss in the i^{th} window. We use exponential averaging to estimate next loss. Suppose we are currently at the n^{th} window, p_n^* is determined by

$$p_n^* = \left\lceil \alpha.p_n + (1-\alpha).p_{n-1}^* \right\rceil$$

In this experiment, we have picked $\alpha = \frac{1}{2}$. This value turns out to work just fine, as shown in section 5. Whether or not there exists an optimal value for α is subject to further investigation. Basically, α measures how much weight we would like to give to the current network status. The larger α is, the less weight we give to the history of network behavior. p_n^* is rounded up because we want to assume the worse error.

4.2 Illustration of the protocol

Figure 3 illustrate an example of how client and server interact. $< j, \pi_i >$ is the time where server sends the i^{th} frame of the j^{th} buffer window. ACK_j containing the estimated p_j^* sent back by the client. By the time server gets ACK_j, it could be in the $(k-1)^{th}$ buffer window. So, it uses ACK_j for the k^{th} buffer window. Lastly, ACK_{j-1} is lost, so we have not used it for transmission of any of the buffer window subsequently.

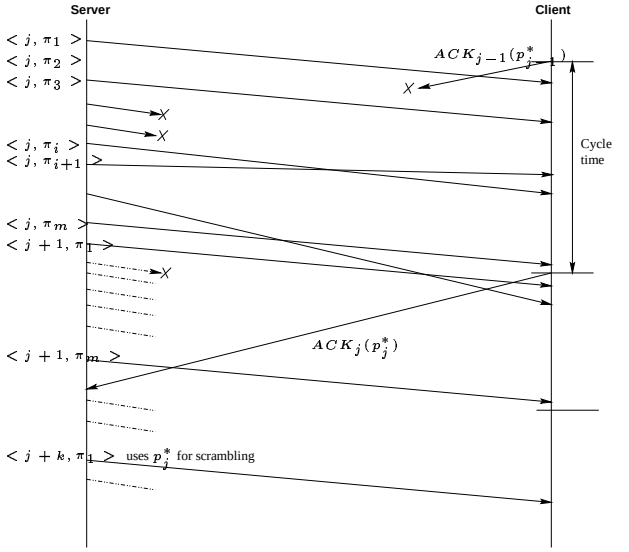

Figure 3: A sample session

5 Experimental Evaluation

The following two sections presents the evaluation of our scheme in two scenarios. In one case the protocol described in section 4.1 has been implemented and tested over a long haul network. In the second case, we use a data set extracted from a real-time application such as *Internet Phone* and simulate our protocol. We show the reduction in CLF in both the cases. Our protocol has smoothed out CLF to be within the range of perceptually acceptable tolerance. Also, it adapts quite well with abnormality in network loss pattern. Moreover, *almost all* of CLF values are within the range of perceptual tolerance (see section 2.1. Thus this approach of of using End User QoS as a direct means to control Media Delivery shows a lot of promise. There are a number of extensions to the protocol which we have been and are currently looking into. These are briefly discussed in section 6;

5.1 Video Experiment 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 [2]. The experiment was conducted at 9:45am when network traffic is expected to be average. Both clips have resolution 512×384. Clip 1 frame sizes varies from 5276 to 36364 bytes with 9544 bytes as the median, 10845 bytes as the mean and 4450.7 is the standard variation. These numbers for clip 2 respectively are 5072, 34408, 10282, 10916 and 3642.8. Clip 1 contains 2607 frames and clip 2 contains 1736 frames. 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  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 4 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).

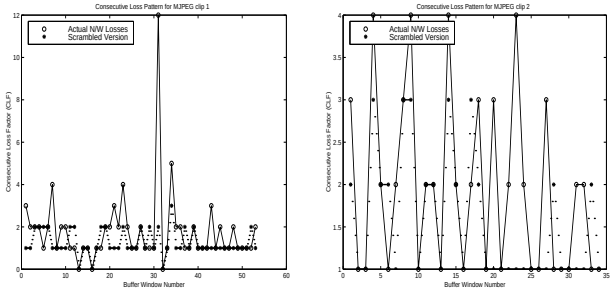

Figure 4: Video transmitted over a long haul network

5.2 Simulation: Using Internet Phone data

The data [3] 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* are both SUN UltraSparc 1 workstations, 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] Thanks to Mr. Thanh C. Nguyen at the Department of Computer Science, University of Wisconsin, Milwaukee for helping us in conducting this experiment

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

2 pm. The two files presented here are of voice packets of sizes 160 and 480 bytes. As can be seen from figure 5, 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).

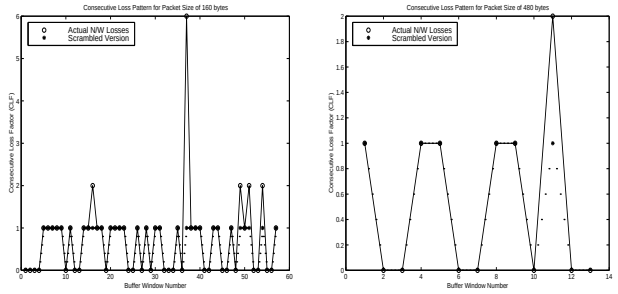

Figure 5: Real-time application such as Internet Phone

6 Concluding Remarks

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 idea of *error spreading*, which takes spots of concentrated bursty losses and spreads it evenly over the entire stream. This makes the stream more acceptable from a perceptual viewpoint [6]. Our experiments over the Internet show that the scheme is quite effective.

Our ongoing work is addressing a number of issues. First, we want to develop an analytical formulation for the unbounded network error case. Second, we want to develop more sophisticated scrambling techniques, and evaluate them through analysis and experimentation. Thirdly, modeling of streams which have inter-frame dependency, such as MPEG needs to be done. Finally, we want to extend this idea to groups of synchronized streams.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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*.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

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

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