Multimedia Tools and Applications, 21, 281–305, 2003

Packet Permutation: A Robust Transmission Technique for Continuous Media Streaming Over the Internet

JEN-WEN DING jwding@mail.ksut.edu.tw
SHENG-YUAN TSENG tsy@mail.ksut.edu.tw
YUEH-MIN HUANG raymond@mail.ncku.edu.tw
Department of Engineering Science, National Cheng Kung University, Tainan, 701 Taiwan, Republic of China

Abstract. With the growing popularity of the Internet, there is an increasing demand to deliver continuous media (CM) streams over the Internet. However, packets may be damaged or lost during transmission over the current Internet. In particular, periodic network overloads often result in bursty packet losses, degrading the perceptual quality of CM streaming. In this paper, we focus on reducing the impact of this bursty loss behavior. We propose a novel robust end-to-end transmission scheme, referred to as packet permutation (PP), to deliver pre-compressed continuous media streams over the Internet. At the server side, PP permutes, prior to transmission, the normal packet delivery sequence of CM streams in a specific way. The packets are then re-permuted at the receiver side before they are presented to the application. In this way, the probability of losing a large number of packets within each CM frame can be significantly reduced. To validate the effectiveness of PP, a series of trace-driven simulations are conducted. Our results show that for a given quality of service (QoS) requirement of CM streaming, PP greatly reduces the overhead required by traditional error control schemes, such as forward error correction (FEC) and feedback/retransmission-based schemes.

Keywords: continuous media streaming, video-on-demand, robust delivery scheme, bursty packet losses, packet permutation

1. Introduction

1.1. Motivation

With the rising popularity of the Internet and the dramatic evolution in multimedia and communications technologies, there is a growing demand to support continuous media (CM) streaming over the Internet. With CM streaming [12, 27], CM data can be played as soon as it is delivered to the client. Thus, the user does not have to wait for a complete file to be downloaded before viewing (or listening to) it. In contrast to the conventional "download and play" approach, the streaming approach has the advantages of displaying CM files much sooner and not requiring a large amount of disk space on the client site. In addition, copyright concerns are reduced since a complete copy does not need to reside on the client's hard disk drive.

However, the current Internet is unsuitable for supporting CM streaming since it offers only a single class best effort service [6, 12, 16]. One of the main drawbacks of this simple service model is that packets may be damaged or lost during transmission. In particular,

successive packets are often dropped by routers when the network becomes congested, resulting in bursty packet losses. Previous studies on packet loss on the Internet have indicated that this behavior is primarily caused by the drop-tail queuing discipline employed by many Internet routers [19]. Recently, the random early detection (RED) congestion avoidance algorithm and its variants have been proposed, which may alleviate the bursty loss phenomenon [8, 32]. A RED-based router can keep the average queue size low while allowing occasional bursts of packets in the queue. Nevertheless, the RED algorithm is designed primarily to accompany transport protocols that can respond to congestion indications, such as TCP. It is unsuitable for transport protocols like UDP, although in practice most CM data like video and audio employ UDP for its greater efficiency [12, 26, 27]. Another problem with RED is that RED-based routers have not been broadly deployed in the Internet. As a result, to date, bursty losses still remains a troublesome problem [2, 4, 20, 33].

Previous studies on CM viewing have shown that packet losses as well as other problems, such as varying transmission delay and bandwidth, have annoying effect on the perceptual quality of CM data. In particular, since numerous CM compression algorithms apply inter-frame compression techniques [9], the effect caused by packet losses may be significant. With inter-frame dependancy, losing any packet of an intraframe (also known as a reference frame or an anchor frame) may result in the loss of all interframes related to that intraframe. To overcome the deficiency of the Internet, two types of approaches have been proposed. The first, known as the *network layer reservation* approach, augments the best-effort service provided by the Internet with other services that offer various degrees of quality of service (QoS) guarantees. This approach requires the implementation of specific resource allocation and reservation mechanisms within the network. Much research has been devoted to this issue over the past few years. Examples include the development of Asynchronous Transfer Mode (ATM), Internet Stream Protocol Version 2 (ST-II), Resource ReSerVation Protocol (RSVP), IP Version 6 (IPv6), and Differentiated Services (Diffserv) [12, 15, 16, 29]. Unlike the first type of approach, which provides QoS at either the data link layer (such as with an ATM) or at the network layer, the second type of approach, known as the *application layer framing* approach, adds specific functions to applications to enable them to adapt to the uncertainties of the best-effort service [5, 12]. For instance, applications can adapt to varying available bandwidth by adjusting the frame rate or resolution of CM data [3, 12]. They can also adapt to varying transmission delay by employing a small buffer at the receiver side to compensate for network delay jitter [12, 27]. Since the newly developed resource reservation mechanisms have not been broadly deployed in the Internet, in the near future, the second type of approach will be a quicker and more practical solution to deliver CM streams over the Internet [5, 12].

In this paper, we focus on reducing the annoying effect caused by bursty packet losses. We propose a novel robust end-to-end delivery scheme for pre-compressed CM streams over the Internet. This delivery scheme, termed packet permutation (PP), is as an *application layer framing* approach. At the server side, PP permutes the normal packet delivery sequence of CM files prior to transmission. Specifically, PP interleaves the packets from distinct frames prior to transmission. At the client side, PP re-permutes the received packets back to the original delivery sequence before they are presented to the application. Therefore, as will be explained in Section 3, when a bursty loss occurs, the probability that a frame loses a large

number of packets is significantly reduced. This effect is important since it implies that the overhead required to recover the lost packets of all reference frames of a CM stream to achieve a predefined QoS can be greatly reduced by PP. The effectiveness of PP is validated via a series of trace-driven simulation experiments, which we shall discuss in Section 4.

1.2. Related work

Conventional error control schemes employed by CM applications can be classified into two categories: redundancy-based and non-redundancy-based. Redundancy-based schemes include proactive and reactive schemes. Proactive schemes take some kind of action to protect data in advance. A widely used proactive scheme is forward error correction (FEC), which encodes repair packets from normal data packets and sends them together to the receiver. Reactive schemes take action to recover damaged or lost data when a transmission error has been detected. A widely used reactive scheme is automatic repeat request (ARQ), which retransmits damaged or lost packets according to the feedback information sent by the receiver. Examples of CM applications employing FEC, feedback/retransmission-based error control schemes, or their hybrid can be found in [1, 3, 23]. In [10, 34], Ghanbari, Seferidis, and Zhu propose non-redundancy-based error control schemes. They employ spatial and temporal interpolation (also known as error concealment) to repair the errors caused by damaged or lost packets. Spatial interpolation reconstructs a missing piece in a CM frame from its adjacent non-missing regions in the same frame, while temporal interpolation replaces the missing region with the corresponding region in a previous frame. An important feature of PP is that it is orthogonal and complementary to these conventional error control schemes.

To disperse the effect of packet losses, Perkins proposed a simple audio frame interleaving technique, which is well known in the area of error resilient audio transmission [21]. Unlike PP, this work is targeted primarily at stored audio streaming and does not take into account the inter-frame dependency adopted by most CM compression algorithms.

To relieve the impact caused by bursty packet losses on the Internet, Varadarajan et al. have recently proposed several error spreading algorithms for CM streaming [17, 30, 31]. Unlike PP, their work permutes the delivery sequence of CM frames, rather than that of CM packets. Their intention is to reduce the maximum number of consecutive lost frames, referred to as consecutive loss factor (CLF), caused by bursty packet losses. However, the unit of bursty packet losses is a packet, rather than a CM frame. The size of a CM frame is typically much larger than that of a packet, and different types of frames have very different sizes. For example, for MPEG-1 video, the typical size of an *I* frame is approximately three times that of a *P* frame and approximately seven times that of a *B* frame. An *I* frame can be divided into approximately 37 packets of 576 bytes (64 bytes IP header and 512 bytes data), the Internet standard maximum transmission unit (MTU) [22]. On the other hand, the length of a bursty loss observed by an end-to-end connection generally ranges from two to six packets in the current Internet [2, 33]. Thus, the probability that many consecutive CM frames are damaged by a bursty loss is very low. As a result, although "frame permutation" alleviates the worst case (i.e., reducing the maximum number of consecutive lost frames), the probability distribution functions of packet losses within each CM frame remain essentially

unchanged. As will be shown in Section 4, changing the frame delivery sequence fails to reduce the overhead required to recover the lost packets of reference frames (or anchor frames) to achieve a predefined QoS. Another problem with their work is the need for the upper bound on the size of a bursty loss, which is difficult to obtain in reality.

1.3. Research contributions

Due to the inter-frame dependency of CM data [9], bursty packet losses may significantly degrade the perceptual quality of CM streaming. Hence, addressing this issue is critical for many CM streaming applications, such as video-on-demand, distance learning, etc. PP takes into account the inter-frame dependency adopted by most CM compression algorithms, and interleaves the packets from significant and non-significant CM frames accordingly, in order to spread the errors out over significant and non-significant CM frames, thereby reducing the annoying effect caused by bursty packet losses. Unlike previous error spreading techniques for CM streaming [17, 30, 31], PP takes the inter-frame dependency of CM streams into account and employs a quite different permutation algorithm. More importantly, it permutes the data of CM files at the packet level, rather than at the CM frame level. In this way, PP can greatly reduce the probability of losing a large number of packets within each CM frame. As will be shown in Section 4, while previous work failed to change the probability distribution functions of packet losses within each CM frame, PP effectively reshapes these distributions to reduce the probability of losing a large number of packets within each CM frame. Consequently, while PP can significantly reduce the overhead required by traditional error recovery schemes to achieve a given QoS, frame permutation fails to reduce this overhead.

In addition to being effective, PP has four salient features. First, it is orthogonal and complementary to conventional error control schemes such as FEC and feedback/retransmission-based schemes. It is worth noting that since PP can work synergistically with FEC, it is very suitable for CM applications using multicast. Second, PP can be applied to many types of CM formats with inter-frame dependency. To illustrate PP, however, we primarily apply it to MPEG-1 video, a prevalent video compression format, in this paper. The generic version of PP algorithm is given in Section 3.2. Third, PP can be easily combined with various rate control algorithms, which are important for CM streaming applications (see Section 3.4) [3, 12]. Fourth, PP has very low computational complexity, which makes the implementation practical.

The rest of this paper is organized as follows. Section 2 reviews the important features of MPEG and packet-level FEC that are relevant to this paper. Section 3 details PP and Section 4 presents the results of the trace-driven simulations. Finally, Section 5 concludes this paper.

2. Preliminaries

2.1. General description of MPEG

MPEG-1 is a standard intended for lossy video compression and decompression with quality comparable to that of VHS video cassette recorders [9]. MPEG-1 standards use inter-frame

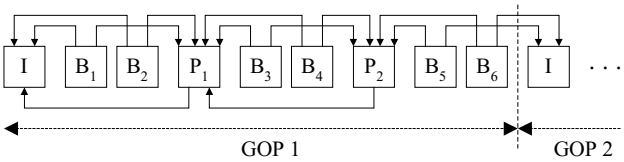


Figure 1. MPEG frame dependency (arrows indicate dependencies).

NPS: I B B P B B P B B I B B P B B P...
NDS: I P B B P B B I B B P B B P B B...

Figure 2. Normal playback sequence (NPS) and normal delivery sequence (NDS).

encoding to reduce temporal redundancy and intra-frame encoding to reduce spatial redundancy. Typically, the coded rate of MPEG-1 compressed video is about 1.5 Mbps. MPEG-1 classifies the frames in a video stream into three types: intrapictures (*I* frames), predicted pictures (*P* frames), and interpolated pictures (*B* frames). The names imply the type of compression each frame type undergoes. The encoder uses only spatial redundancy to compress *I* frames. The error due to lossy compression for an *I* frame is small since no inter-frame encoding is used. *I* frames thus have high picture quality at the expense of moderate compression ratios. *I* frames serve as reference points for inter-frame encoding for *P* and *B* frames. *P* frames use inter-frame encoding based on motion prediction with reference to a previous *I* or *P* frame. This technique yields better compression ratio, but the error due to the lossy compression is higher. *B* frames provide the highest compression ratio by interpolating the difference between a past and a future reference frame (*I* or *P*). *B* frames never serve as reference. Figure 1 shows the relationship among the three types of frames. Video compressors usually produce MPEG-1 streams according to a user-specified pattern called GOP. A GOP contains a fixed number of consecutive frames and guarantees that the first picture is an *I* frame. For instance, a commonly used GOP is "IBBPBBPBBPBBPBB." GOP patterns are used to determine the tradeoffs between quality, compression ratio, and flexibility regarding random access requirements. In delivering MPEG-1 streams to receivers, reference frames are transmitted first since they are required to decode other frames. Hence, the normal delivery sequence (referred to as NDS) and the normal playback sequence (referred to as NPS) differ, as illustrated in figure 2. It is worth mentioning that MPEG-2 standard, which is built on MPEG-1 standard and can support HDTV quality, also classifies video frames into *I* frames, *P* frames, and *B* frames since it also employs inter-frame encoding to reduce temporal redundancy and intra-frame encoding to reduce spatial redundancy.

2.2. FEC and feedback/retransmission

FEC and feedback/retransmission-based schemes are two key error control mechanisms used in many applications and communication protocols. However, feedback/retransmission-based schemes have several drawbacks when applied to CM data

communication. First, the time needed to detect damaged or lost packets is undetermined and may be long, and the time needed to retransmit damaged or lost packets varies depending on the conditions of the network at the time of retransmission. These factors may introduce a considerable amount of delay jitter [6, 12]. Second, feedback/retransmission-based schemes are not suitable for multicast environments. The feedback information (such as ACK or NACK) may lead to a situation known as feedback implosion: when the number of receivers in a multicast group increases, the sender is forced to process more and more messages from receivers that either indicate correct data reception or request retransmission [12]. Although a number of solutions have been proposed to avoid this implosion effect, such as randomized timers, local recovery, and hierarchical recovery, they result in unpredictable delays, making them unsuitable for CM applications [12]. Unlike feedback/retransmission-based schemes, FEC does not suffer from these problems, although at the expense of some extra bandwidth. In view of this, we mainly consider the conjunction of PP and FEC in this paper. In the following we describe the important features of FEC that are relevant to this paper.

Error correcting codes (e.g., Reed-Solomon codes) were initially applied in domains where bits could be erroneous or missing, but have more recently been applied to repairing packet losses at the network layer. Packet-level FEC works as follows. The sender forms blocks, where each block consists of a subset of the data packets it wishes to deliver reliably. The number of data packets that are used to form a block is commonly referred to as the block size, denoted by k. The sender inputs the k packets into its FEC encoder which then generates repair packets for that block. A receiver uses its FEC decoder to recover the k data packets from any combination of k distinct packets that are either the data packets or the repair packets of that block. Figure 3 shows an example, where a sender groups six data packets into a block, encodes two repair packets from this block, and transmits all eight packets to the receiver. As soon as the receiver receives any six distinct packets related to the block (in this example, five data and one repair), it activates the decoder and recovers the lost data packets. It is worth mentioning that FEC requires very little computational effort to implement. We refer the reader to [14, 18, 24] for details of Reed-Solomon codes and packet-level FEC techniques.

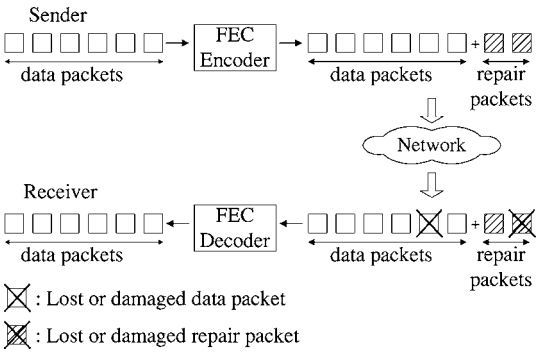


Figure 3. An example of packet level-FEC.

3. Packet permutation

This section first examines the impact caused by bursty packet losses on CM streaming. Then, it details how PP reduces this impact. Finally, the combination of PP and some application layer framing approaches is given.

3.1. Impact of bursty packet losses

Consider transmitting an MPEG-1 video stream over the Internet in accordance with the normal delivery sequence (NDS) discussed in Section 2. Whenever a bursty loss occurs in the stream (i.e., successive packets of the stream are dropped), three types of situations may occur for the *I* frames of the stream. First, if the bursty loss occurs on other types of frames, then no *I* frame packet will be dropped, as shown in figure 4(a). Second, if the bursty loss occurs across an *I* frame and its following (or preceding) frame, then a few *I* frame packets will be dropped, as shown in figure 4(b). Third, if the bursty loss occurs within an *I* frame, then successive *I* frame packets will be dropped, as shown in figure 4(c). From the discussion in Section 2, it is known that losing any of the *I* frame packets may result in a significant impact on the visual quality of the stream since the loss of an *I* frame may render the whole GOP related to that *I* frame useless. A GOP may contain 12 or 15 frames or hundreds of packets depending on its pattern and size. Hence, it is vital to recover all lost packets for *I* frames. However, this is not an easy task. If FEC is employed, because of the unpredictable behavior of bursty losses, in order to receive all *I* frames correctly, each *I* frame must use a large enough redundancy (i.e., repair packets) in order to recover from the worst case created by bursty losses (see figure 4(c)). If feedback/retransmission-based schemes are employed, there may exist some situations where so many packets of an *I* frame are lost such that it is difficult to retransmit them on time. Alternatively, a more cost

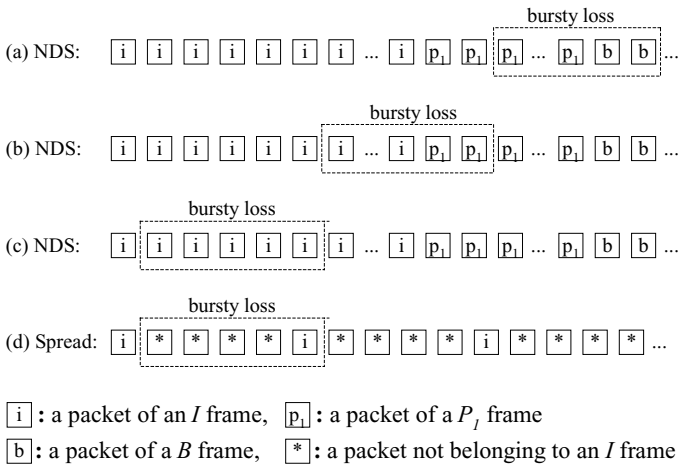


Figure 4. Impact of a bursty loss on *I* frames in an MPEG stream.

effective approach is to provide a statistical QoS guarantee. Specifically, let Q denote a QoS parameter representing a percentage (e.g., 99%) of GOPs. The statistical approach provides a service that ensures that the I frames of at least Q percent of all GOPs can be received correctly. With statistical QoS guarantees, the redundancy required by FEC, or the number of retransmitted packets required by feedback/retransmission-based schemes, can be effectively reduced at the expense of a slight degradation in the perceptual quality of the stream.

Observe from figure 4(b) and (c) that if the distance between any two consecutive I packets can be increased, then the number of I packets dropped in a bursty loss will be reduced. The longer the distance, the less the loss. A direct method to achieve this goal is to spread all I packets across the entire delivery sequence, as shown in figure 4(d). In this way, we can eliminate many ill conditions incurred by bursty losses (e.g., the situations shown in figure 4(b) and (c)). In other words, the number of I packets dropped in a bursty loss can then be reduced to a small number. Although this method reduces the probability of losing larger numbers of I packets, it also increases the probability of losing smaller numbers of I packets. This is because when all I packets are spread out, an I packet (or a few I packets) is more likely to encounter a bursty loss. Therefore, spreading all I packets out reshapes the probability function of the number of lost packets within each I frame as depicted in figure 5. (For simplicity, the probability functions in figure 5 are depicted as continuous, while in reality, since the number of lost packets must be an integer, they are discrete.) Obviously, the reshaped function is better. If feedback/retransmission-based error control schemes are adopted, it virtually eliminates events in which so many I frame packets are lost that they cannot be retransmitted on time. If FEC error control scheme is adopted, it substantially reduces the redundant information required to protect all I frames. Figure 5 also conceptualizes this reduction in redundancy, where $R_{N,99\%}$ and $R_{S,99\%}$ denote the redundancy required by NDS and by the delivery sequence with spread out I packets for $Q = 99\%$, respectively.

Similarly, it is desirable to reduce the probability of losing larger numbers of packets for P_1 frames since the loss of a P_1 frame will render the subsequent P frames and all B

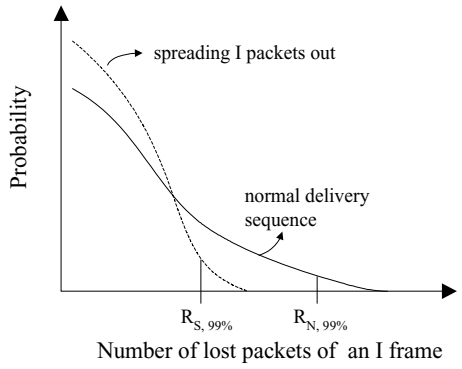


Figure 5. The probability function of the number of lost packets of an I frame.

frames within the same GOP useless. Therefore, it is also desirable to spread all packets of P_1 frames across the entire delivery sequence. A similar argument applies to the remaining reference frames. Actually, increasing the distance between any two consecutive packets belonging to the same reference frame is the main idea behind PP.

3.2. Packet permutation algorithm

PP uses a data channel (e.g., a UDP connection) for sending data packets of CM streaming, and a control channel (e.g., a TCP connection) for communicating the control information required by PP and error control schemes. At the server side, in order to spread the data packets of each frame type across the entire delivery sequence, PP interleaves the packets from distinct frames. The size of a packet is set to the Internet standard MTU, 576 bytes [22]. PP permutes the normal packet delivery sequence of one or several GOPs at a time before transmitting them to the client. The number of GOPs permuted at a time is an adjustable parameter (denoted by G), which can be determined by both the server and the client via an initial negotiation before streaming. Later we shall discuss the effects of this parameter. At the client side, the permuted packets are received and temporarily stored in a buffer maintained by PP. Whenever the reception of G GOPs is completed, the packets are re-permuted back to their original delivery sequence and then passed to the application. For simplicity of presentation, we first consider the case where packet permutation is carried out individually for GOP's. Figure 6 illustrates the concept of streaming with PP. In this case, PP alters the packet delivery sequence within each GOP, but preserves the order in which distinct GOPs are sent. The delivery sequence generated by PP is referred to as packet permutation sequence (PPS). Let i_k^g denote the kth packet of the I frame in the gth GOP, $p_{j,k}^g$ denote the kth packet of the P_j frame in the gth GOP, and b_k^g denote the kth packet of the B frames in the gth GOP (since B frames are non-reference frames and losing them do not affect any other frames, b_k^g is not indexed by j). Figure 7 illustrates how the PPS of a GOP is generated (with the abbreviation of the corresponding superscripts).

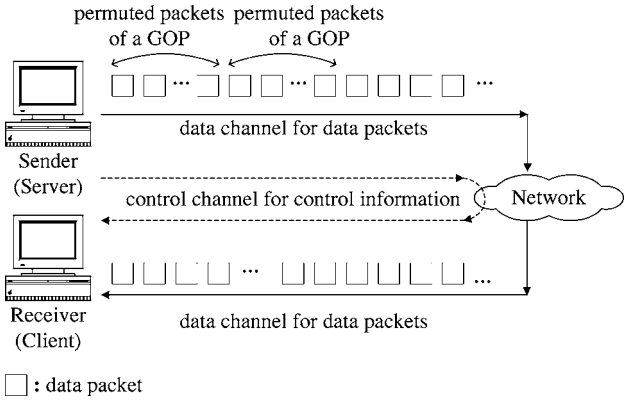


Figure 6. An example of streaming with one GOP permuted at a time.

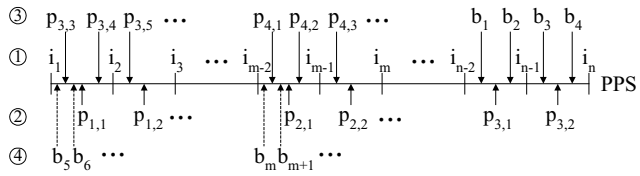


Figure 7. Generating packet permutation sequence for a GOP.

(a) PPS of one GOP (GOP Pattern : IBBPBBPBBPBB) :

$$i_1 \to b_1 \to p_{1,1} \to b_2 \to i_2 \to b_3 \to p_{1,2} \to b_4 \to i_3 \to b_5 \to p_{2,1} \to b_6 \to i_4 \to b_7 \to p_{22} \to b_8 \to i_5 \to p_{3,1} \to i_6 \to p_{3,2} \to i_7$$

(b) PPS of two GOPs (GOP Pattern : IBBPBBPBB) :

$$i_1^1 \to b_1^2 \to p_{1,1}^1 \to b_2^1 \to i_1^2 \to b_2^2 \to p_{1,1}^2 \to b_3^1 \to i_2^1 \to b_3^2 \to p_{1,2}^1 \to b_4^1 \to i_2^2 \to b_4^2 \to p_{2,1}^1 \to b_5^1 \to i_3^1 \to b_5^2 \to p_{2,1}^2 \to b_6^1 \to i_3^2 \to b_6^2 \to p_{2,2}^1 \to i_4^1 \to p_{2,2}^2 \to i_4^2 \to b_1^1 \to i_5^2$$

Figure 8. The PPS of one GOP and two GOPs.

Initially, PP adds the most significant frame packets (i.e., I frame packets) to PPS. Then, it inserts each of the second most significant frame packets (i.e., P_1 frame packets) between every two consecutive I packets. Next, it similarly inserts the third most significant frame packets (i.e., P_2 frame packets) into the other consecutively arranged I packets. Other frames are similarly inserted based on their significance. After there is a packet inserted between every two consecutive I packets, the insertion operation returns to the beginning of the PPS and then continues. A similar process is repeated until all packets have been added to the PPS. Note that since PP inserts packets in the order of their importance, first, the largest distance is between consecutive I packets, and the next largest distance is between consecutive P_1 packets, and so on. Consequently, the impact caused by bursty losses on I frames is the smallest, that upon P_1 frames is the next smallest, and so on. For a simplified discussion of PP, consider the following example. Assume that the pattern of a GOP is "IBBPBBPBBPBB" and that the numbers of packets of I, P_1, P_2, P_3, and all B frames are 7, 2, 2, 2, and 8, respectively. Figure 8(a) depicts the PPS of the GOP. Note that since PP collects a group of data, permutes them, and then delivers them, it is only suitable for stored CM streaming (such as VOD), not for live CM streaming (such as video conferencing).

As mentioned above, PP can also permute the packets from two or more GOPs at a time. The advantage of permuting several GOPs at a time is that the distance between consecutive packets belonging to the same reference frame can be extended even more. The drawback is that this may incur a slightly longer initial delay and a larger buffer requirement. The additional delay introduced by permuting an additional GOP is approximately 0.5 seconds

(if a GOP contains 15 frames and the playback rate is 30 frames/second); and the additional buffer space needed is approximately 76 KB (if the average bit rate is 1.2 Mbps). Fortunately, such extra delays and buffer space are acceptable for most applications. The way multiple GOPs are permuted is similar to that discussed above. The main difference is that the frames of the same type in distinct GOPs are merged and treated as a single frame with their packets interleaved with each other. To better explain this idea, consider the following example. Assume that two GOPs are permuted and the GOP pattern is "IBBPBBPBB." Assume that for the first GOP, the numbers of frame packets of I, P_1, P_2, and all B's are 4, 2, 2, and 6, respectively. For the second GOP, the numbers of frame packets of I, P_1, P_2, and all B's are 5, 1, 2, and 6, respectively. Figure 8(b) shows the PPS of the two GOPs. Note that in figure 8(b), the next I packet after i_1^1 is i_1^2; the next P_1 packet after $p_{1,1}^1$ is $p_{1,1}^2$; and so on. As a result, the distance between any two consecutive packets belonging to the same reference frame is even larger, which increases the robustness of the delivery sequence. Formally, figure 9 gives the generic version of PP algorithm at the sender side, which can be applied to other CM formats with inter-frame dependency. In figure 9, the term elementary CM structure refers to the basic group of data that can be permuted by PP at a time (e.g., a GOP of MPEG-1 video). It can be seen from figure 9 that the computational complexity of PP is $O(T \cdot G \cdot m)$, where T is the number of frame types in an elementary CM structure for permutation, G is the number of elementary CM structures for permutation at a time, and m is the maximum number of packets in a frame. This low computational complexity makes the implementation practical.

After deciding PPS, packets can be sent according to PPS. To facilitate the re-permutation work at the client side, the PP algorithm at the server side adds two fields in each data packet prior to transmission: (1) *GOP id* and (2) *packet id*. The *GOP id* is a sequence number used to identify a GOP in a stream; the *packet id* is a sequence number used to identify a packet in a GOP. With these two fields, whenever a client completes the reception of G GOPs, the PP algorithm at the client side can generate a read sequence indicating the normal order in which the data packets are read. Then it strips off the added two fields in each data packet and then passes them to the application according to the read sequence. Figure 10 gives the generic version of the PP algorithm at the receiver side, where *ECMS* is the abbreviation of "elementary CM structure" for permutation. The sorting task in figure 10 can be implemented in numerous ways, e.g., using Quicksort algorithm [13] to sort $A[j]$, $j = 1, 2, \ldots, N$, in terms of $A[j].ECMSID \times 1000 + A[j].PKTID$ (generally, the number of packets in an *ECMS* is far less than 1000 for most CM formats).

In practice, three additional design issues must be considered in implementing the receiver-side generalized PP algorithm. First, a buffer must be maintained by PP for receiving G ECMS's. Let $Size(ECMS_j)$ denote the size of the jth ECMS. The size of a buffer is set to $\max\{\sum_{j=i\cdot G+1}^{j=(i+1)\cdot G} Size(ECMS_j), i = 0, 1, 2, \ldots\}$, which is given by the sender via the control channel. Second, multiple buffers are necessary since some packets may arrive out of order. Usually, *double buffering* or *triple buffering* is adequate. The actual number of buffers required depends on the network conditions, and we refer the reader to [25] for accurate estimation of the buffer size required at the receiver side. Third, because some packets may be dropped or damaged during transmission, the time for triggering packet re-permutation (i.e., the completion time of the reception of G elementary CM permutation

Algorithm ***Sender-Side Generalized Packet Permutation***($S, G, T, f_1^1, f_1^2, \ldots, f_1^G, f_2^1, \ldots, f_T^{G-1}, f_T^G$)

Input: S (an empty sequence),
G (the number of elementary CM structures for permutation at a time),
T (the number of frame types in an elementary CM structure for permutation),
f_1^1 (the number of packets of the most significant frame in the first elementary CM structure),
f_1^2 (the number of packets of the most significant frame in the second elementary CM structure),
...
f_1^G (the number of packets of the most significant frame in the G^{th} elementary CM structure),
f_2^1 (the number of packets of the second most significant frame in the first elementary CM structure),
f_2^2 (the number of packets of the second most significant frame in the second elementary CM structure),
...
f_T^{G-1} (the number of packets of the T^{th} most significant frame in the $(G-1)^{th}$ elementary CM structure),
f_T^G (the number of packets of the T^{th} most significant frame in the G^{th} elementary CM structure)

Output: S (a sequence indicating the order in which packets are sent)

```
{
  /* packet identification symbol f_{j,k}^i stands for the k^th packet of the j^th most significant frame in the i^th
     elementary CM permutation structure */

  /* Define temporarily used variables */
  int m, k, g, γ, t;

  /* Add the packets of the most significant frames to PPS */
  m = max(f_1^1, f_1^2, ..., f_1^G);
  for (k = 1 to m) {
      for (g = 1 to G) {
          if (f_1^g > 0) {
              add packet identification symbol f_{1,k}^g to the tail of S;
              f_1^g = f_1^g - 1;
          }
      }
  }

  /* Initialize the starting point for insertion */
  γ = 1;
  for (t = 2 to T) {
      /* Add the packets of f_t type frames to PPS */
      m = max(f_t^1, f_t^2, ..., f_t^G);
      for (k = 1 to m) {
          for (g = 1 to G) {
              if (f_t^g > 0) {
                  if (γ = |S|)   /* |S|: the number of elements in S */
                      γ = 1;   /* Return the insertion operation to the head of S */
                  insert packet identification symbol f_{t,k}^g between the γ^th and (γ+1)^th
                      elements of S;
                  γ = γ + 2;
                  f_t^g = f_t^g - 1;
              }
          }
      }
  }
}
```

Figure 9. Generalized packet permutation algorithm at the sender side.

Algorithm ***Receiver-Side Generalized Packet Permutation***(S , B , N)
Input: S (an empty sequence),
B (a buffer containing received packets of G elementary CM permutation structures),
N (number of packets stored in B),
Output: S (a sequence indicating the order in which packets should be read from B)
{
define array $A[N]$;
/* Array $A[N]$ contains N elements with the following three attributes:
$A[j].ECMSID$: used to record the *EMCS id* of the j^{th} packet in B
$A[j].PKTID$: used to record the *packet id* of the j^{th} packet in B
$A[j].POS$: used to record the position of the j^{th} packet in B
*/

/* Initialize array $A[]$ */
for ($j = 1$ to N) {
$A[j].ECMSID$ = *ECMS id* of j^{th} packet in B;
$A[j].PKTID$ = *packet id* of j^{th} packet in B;
$A[j].POS = j$;
}

sort the elements in $A[]$ to meet the following two requirements:

$$\begin{cases} A[j].ECMSID \le A[j+1].ECMSID, & j = 1, 2, ..., N-1 \\ A[j].PKTID < A[j+1].PKTID \text{ if } A[j].ECMSID = A[j+1].ECMSID, & j = 1, 2, ..., N-1 \end{cases}$$

for ($j = 1$ to N)
add $A[j].POS$ to S;
}

Figure 10. Generalized packet permutation algorithm at the client side.

structures) is, in effect, determined by the error control scheme employed. For instance, if packet-level FEC is employed, a timer will be maintained for each buffer in use. If one of the timers expires, re-permutation will be carried out for the appropriate buffer. The value of a timer mainly depends on end-to-end packet delays, which may be periodically measured by sending an end-to-end probe packet on the data channel. We refer the reader to [28] for various approaches for measuring end-to-end packet delays.

3.3. Conjunction of PP and FEC

PP can be easily used in conjunction with FEC. PP handles the repair packets generated for each reference frame as part of each reference frame. Figure 11 depicts this concept. To better understand how this works, consider the GOP in figure 8(a). If the FEC encoder encodes 2, 1, 1, and 1 repair packets for I, P_1, P_2, and P_3 frames, respectively, then the PPS of the GOP will be the one depicted in figure 12, where r_X denotes the repair packet for X frame. It is important to note that, depending on the required QoS, it may not be necessary to encode redundant information for all reference frames. For instance, if the desired QoS is low, the sender may generate repair packets for only I and P_1 frames.

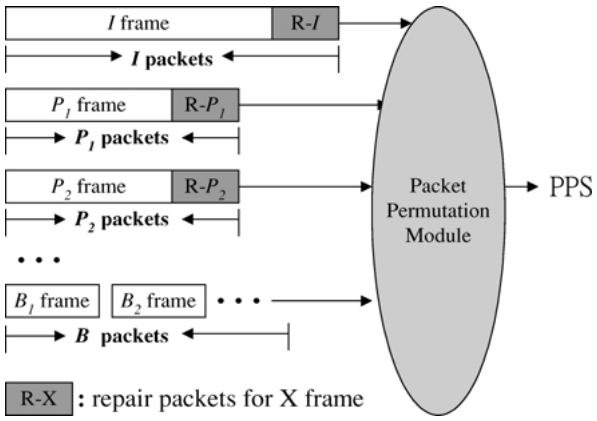


Figure 11. The conjunction of PP and FEC.

PPS of one GOP and FEC repair packets (GOP Pattern : IBBPBBPBBPBB) :

$$i_1 \to r_{P3} \to p_{1,1} \to b_1 \to i_2 \to b_2 \to p_{1,2} \to b_3 \to i_3 \to b_4$$
$$\to r_{P1} \to b_5 \to i_4 \to b_6 \to p_{2,1} \to b_7 \to i_5 \to b_8 \to p_{2,2} \to i_6$$
$$\to r_{P2} \to i_7 \to p_{3,1} \to r_I \to p_{3,2} \to r_I$$

Figure 12. An example of the conjunction of PP and FEC.

3.4. Adapting to varying available bandwidth

The bandwidth provided for a connection by the current Internet varies with time. When a larger number of packets are injected into the network, the bandwidth shared by each connection shrinks accordingly. For CM streaming, if the bandwidth available over the path of the connection is less than the bit rate of the delivered stream, the real-time requirements of the stream will be violated. To overcome this shortcoming, several *rate control algorithms* have been proposed for CM applications [3, 12]. These algorithms usually consist of three functions: (1) periodically collecting the receiving status reported from receivers, (2) determining a proper maximum output rate based on the feedback information, and (3) adjusting the output rate accordingly. PP can easily be combined with these algorithms. Given a rate control algorithm, every time before performing packet permutation, PP selects proper frames from the GOPs to be delivered so that the target output data rate is less than or equal to the maximum output rate determined by the rate control algorithm. The priorities of the selection are as follows: I frames from each GOP have the highest priority, P_1 frames from each GOP have the second highest priority, and so on. Only the packets of the qualified frames are permuted and delivered; the unselected frames are simply discarded.

4. Performance evaluation

4.1. Simulation model

Recent work has shown that the end-to-end packet loss behavior on the Internet can be modeled by a two-state Markov chain, know as the Gilbert model, in which state "1" represents a lost packet and state "0" represents a packet successfully reaching its destination [2, 4, 17, 30, 31, 33]. Figure 13 depicts the Gilbert model, where p denotes the probability of going from state 0 to state 1 and q denotes the probability of going from state 1 to state 0. In our simulation, the Gilbert model was employed to simulate the packet loss behavior on the Internet. Two important parameters of interest regarding this model are the average packet loss rate and the average bursty loss length. According to Markov chain models, the average packet loss rate equals $p/(p+q)$, the limiting probability for state 1. As shown in figure 13, the probability that n consecutive packets are lost is $(1-q)^{n-1}q$. Hence, the residence time in state 1 is geometrically distributed and thus the average length of a bursty loss equals $1/q$. With this model, we can examine the performance of PP under various network conditions. The CM stream tested is a variable-bit-rate (VBR) MPEG-1 file named "cnn.t", whose bit rate trace can be obtained from (ftp://tenet.berkeley.edu/pub/projects/tenet/dbind/traces/) (a few incomplete GOPs in the trace were discarded during simulations). The stream has an average bit rate of 1.2 Mbps with a frame size of 320 × 240, a frame rate of 30 fps, and a GOP pattern of "IBBPBBPBBPBBPBB." Figure 14 shows the frame size distributions

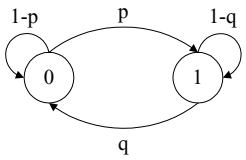


Figure 13. The Gilbert model.

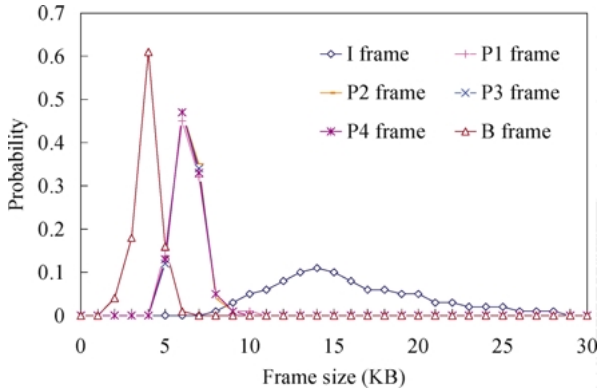


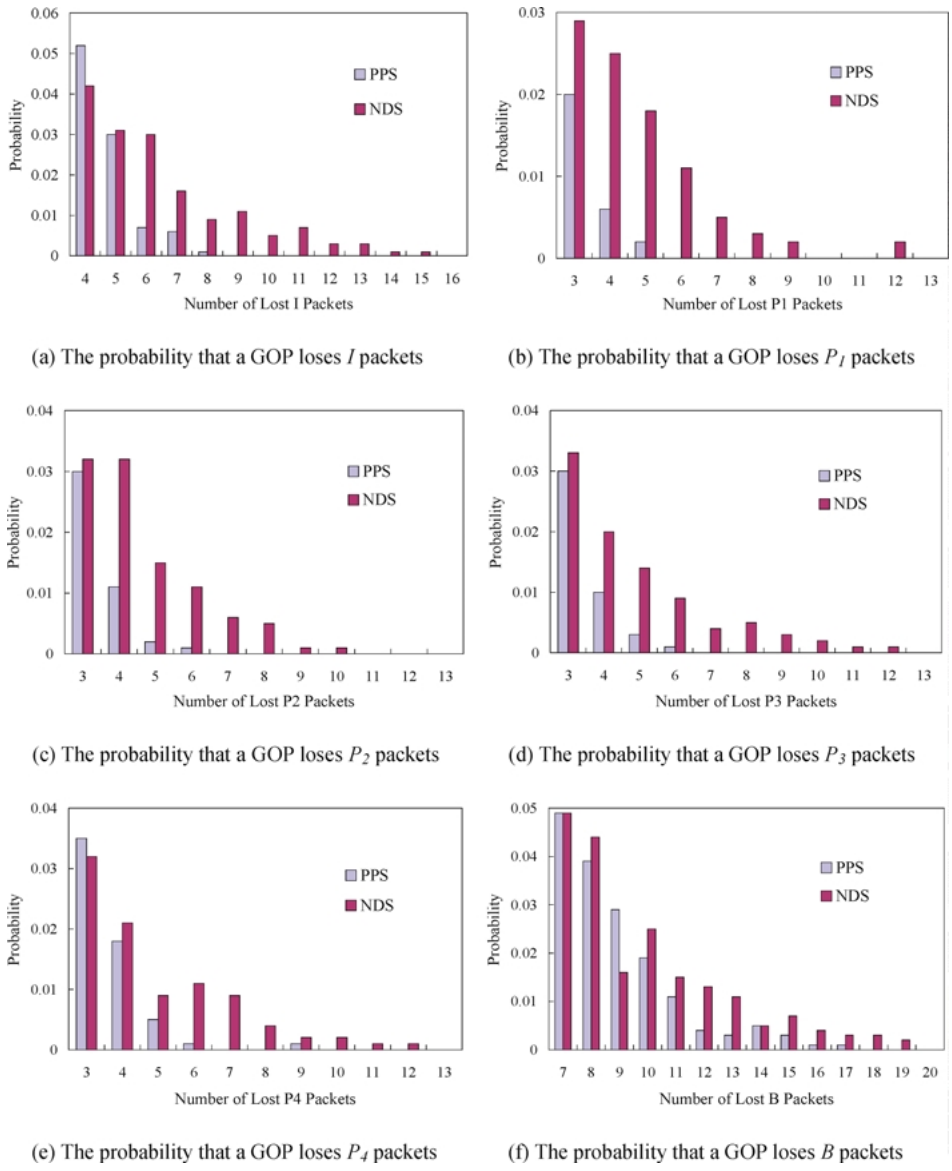
Figure 14. The frame size distribution of the tested video.

of the stream. In each simulation, 2000 GOPs were transmitted with the packet size set to 576 bytes, the Internet standard MTU [22]; that is, about 300,000 packets were transmitted in each simulation. With the simulation model described above, a series of experiments were conducted to compare the performance of the delivery schemes using NDS and PPS under various combinations of network loss, bursty loss length, and the buffer size employed (which is determined by the number of GOPs to be permuted at a time).

4.2. Simulation results

In the first experiment the average packet loss rate was set to 5%, the average number of packets dropped in a bursty loss was set to 3 packets, and the buffer size employed by PP was set to one GOP (more precisely, the buffer size was set to the maximum GOP size). No FEC repair packets were encoded for both the delivery schemes using NDS and PPS. During the process of simulation, the statistics concerning packet losses were maintained to obtain the probability distribution functions of the number of lost packets. Figure 15 shows the probability that a GOP loses packets of each frame type. It can be seen that there is not a consistent reduction in the probability of losing a small number of packets. For example, in figure 15(a) the probability of losing four I packets is actually higher for PPS versus NDS. However, it is clear from the plots in figure 15 that PP reshapes the probability functions so that for each frame the probability of losing larger numbers of packets is significantly reduced, which validates our claim. It can also be observed that the reshaping effect is less significant for B frames. This is because PP assigns the least insertion priority to B packets in generating the PPS, so the distance between consecutive B packets is the smallest compared to that between the packets of other frame types.

In the second experiment, we compared the loss behavior of PPS and NDS for most of the GOPs transmitted. To see the net effect caused by bursty losses, no FEC repair packets were generated for both NDS and PPS (the case with FEC will be studied later). Only the results regarding I frames are shown here because the results for other types of frames are similar. In order to measure performance, we used p-percentile maximum number of lost I packets per GOP; that is, the maximum number of lost I packets within a GOP observed by $p\%$ of GOPs, $0 \leq p \leq 100$, is below this value. Figure 16 plots the 95-percentile and 99-percentile maximum number of lost I packets per GOP versus the average length of a bursty loss under three different packet loss rates: a light loss rate (3%), a medium rate (5%), and a high rate (10%). As can be seen from the figure, first, PP greatly reduces the maximum number of lost I packets in various network conditions. This implies that if FEC is used, PP will effectively reduce the amount of redundant information that is required to achieve a predefined QoS; or if feedback/retransmission-based schemes are used, PP will significantly reduce the events where too many I frame packets are lost so that they cannot be retransmitted on time. Second, the difference between NDS and PPS is especially large when $p = 99$ and the average loss length is long. This is because when $p = 99$ many of the events appearing in the tail of the probability function of the number of lost I packets will be counted in; whereas the probability function of PPS converges much faster than NDS since PPS avoids many situations where a large number of I packets are lost (see figure 15).



(a) The probability that a GOP loses I packets

(b) The probability that a GOP loses P_1 packets

(c) The probability that a GOP loses P_2 packets

(d) The probability that a GOP loses P_3 packets

(e) The probability that a GOP loses P_4 packets

(f) The probability that a GOP loses B packets

Figure 15. The probability that a GOP loses packets for each type of frame.

Figure 17 plots the 95-percentile and 99-percentile maximum number of lost I packets per GOP versus the buffer size employed by PP for different bursty loss lengths: a short length (three packets), a medium length (five packets), and a long length (seven packets). As shown in figure 17, increasing the buffer size (i.e., increasing the number of GOPs permuted at a time) tends to decrease the maximum number of lost I packets for PPS. This

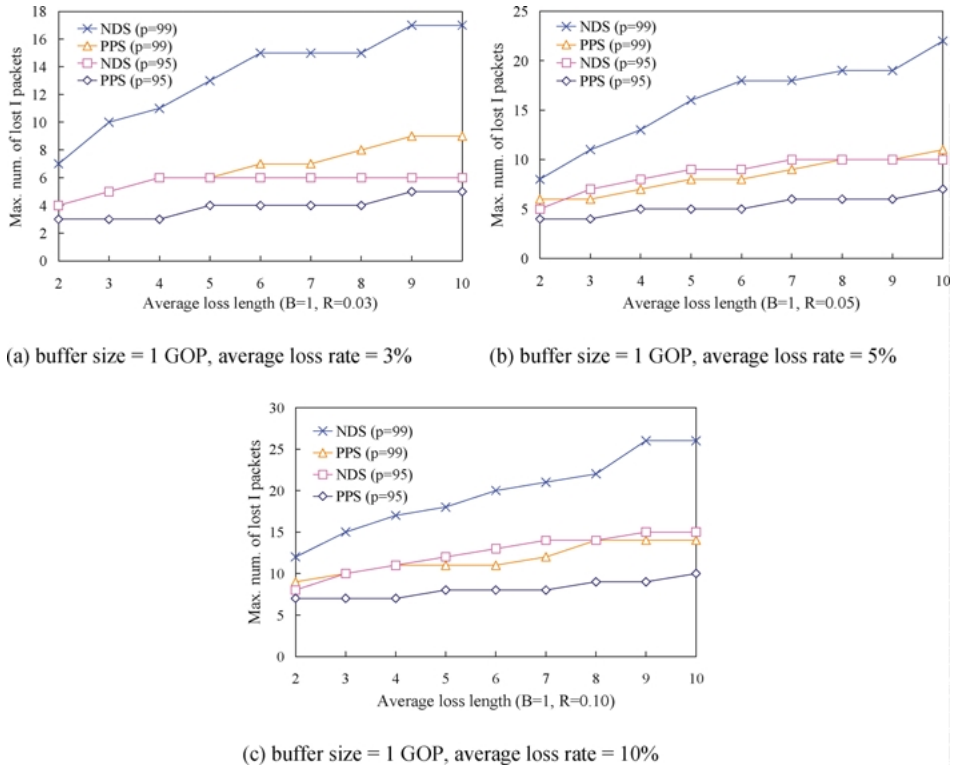
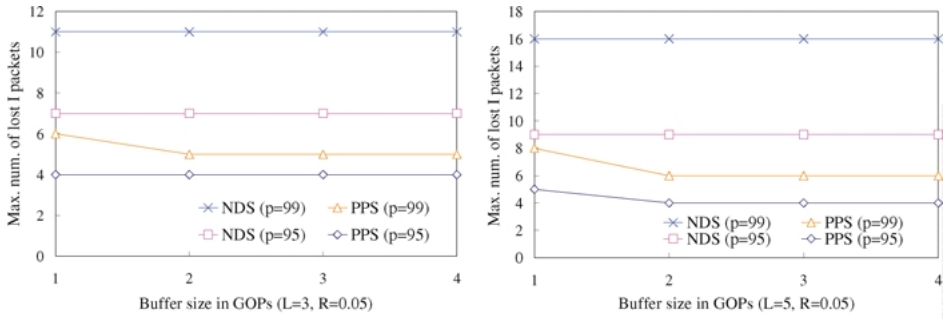


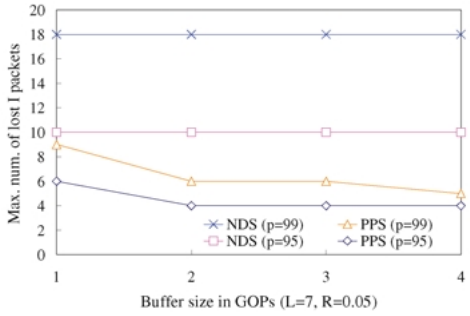
Figure 16. 95-percentile and 99-percentile maximum number of lost *I* packets per GOP versus average length of a bursty loss.

is because increasing the number of GOPs permuted at a time increases the distance between consecutive *I* packets in the PPS, thereby reducing the impact caused by bursty losses (as explained in Section 3.2). However, increasing the buffer size does not necessarily reduce the maximum number of lost *I* packets. For example, as shown in figure 17(a) and (b), when the average loss length is three packets or five packets, using a buffer size larger than two GOPs does not gain any improvement in performance. This is because when the buffer size is set to two GOPs, in most cases, the distance between consecutive *I* packets can reach up to seven packets, which is larger than the average length of a bursty loss. Therefore, system designers should choose a proper buffer size that depends on the bursty loss length of the network in use. For the current Internet, the length of a bursty loss usually ranges from two to six packets [2, 33]. Hence, it may be a good choice to employ a buffer size of two GOPs.

Figure 18 plots the 95-percentile and 99-percentile maximum number of lost *I* packets per GOP versus the average packet loss rate for two average loss lengths, three packets and five packets. The buffer size employed is set to two GOPs. Figure 18 indicates that, for both NDS and PPS, the maximum number of lost *I* packets increases as the average loss rate increases (certainly, the maximum number of lost packets of PPS is much smaller than that

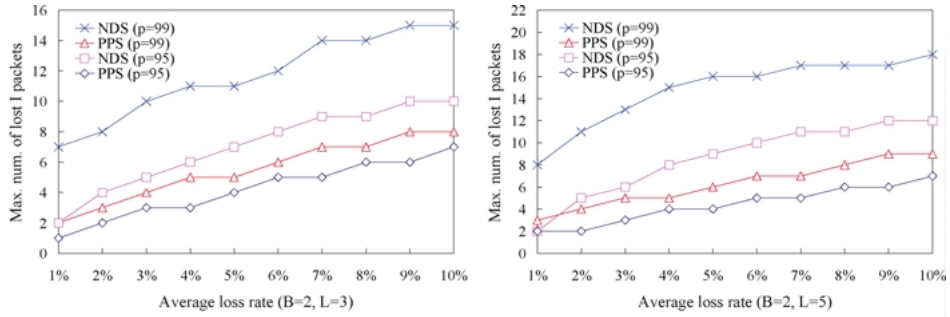


(a) avg. loss length = 3 packets, avg. loss rate = 5% (b) avg. loss length = 5 packets, avg. loss rate=5%



(c) avg. loss length = 7 packets, avg. loss rate = 5%

Figure 17. 95-percentile and 99-percentile maximum number of lost *I* packets per GOP versus buffer size employed by PP.



(a) buffer size= 2 GOPs, avg. loss length= 3 packets (b) buffer size = 2 GOPs, avg. loss length= 5 packets

Figure 18. 95-percentile and 99-percentile maximum number of lost *I* packets per GOP versus average packet loss rate.

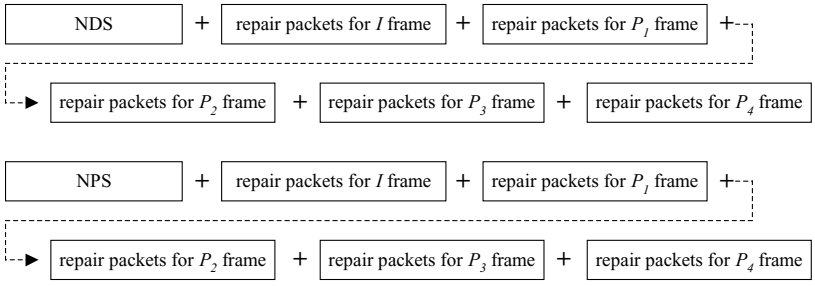
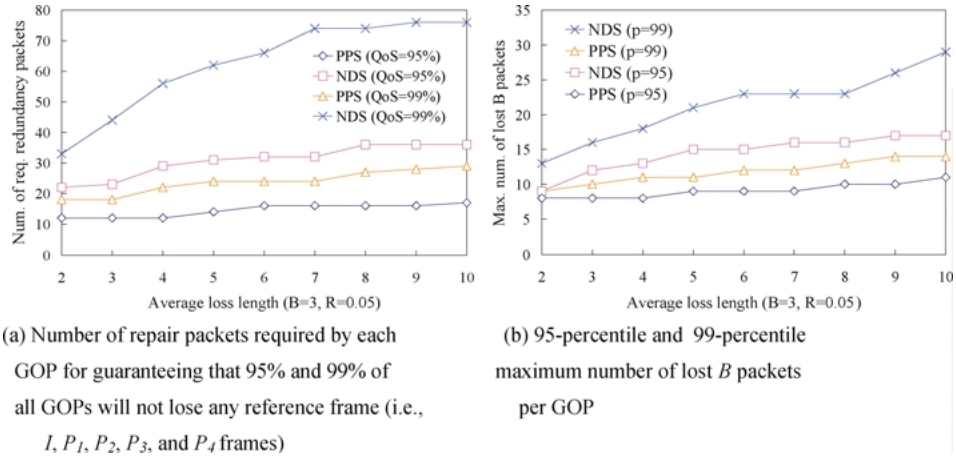


Figure 19. Applying FEC to NDS and NPS.



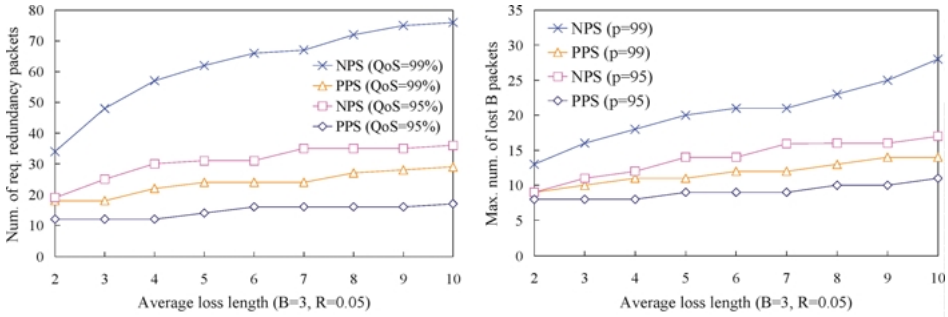
(a) Number of repair packets required by each GOP for guaranteeing that 95% and 99% of all GOPs will not lose any reference frame (i.e., I, P_1, P_2, P_3, and P_4 frames)

(b) 95-percentile and 99-percentile maximum number of lost B packets per GOP

Figure 20. Performance comparison of NDS and PPS for given QoS requirements (buffer size = 3 GOPs, average loss rate = 5%).

of NDS). This result can be anticipated since PP is designed primarily to reduce the damage caused by bursty losses, rather than that caused by loss rate.

In the third experiment, we encoded FEC repair packets for all reference frames in each GOP, and compared the number of repair packets required by NDS and PPS to achieve a particular QoS. The way in which the repair packets are inserted into the PPS has been explained in Section 3.3. The way in which the repair packets are inserted into the NDS is depicted in figure 19. Figure 20(a) shows the number of repair packets required by NDS and PPS for guaranteeing that 95% and 99% of the GOPs transmitted will not lose any of the reference frame packets (i.e., I, P_1, P_2, P_3, and P_4 packets). Figure 20(b) shows the corresponding maximum number of lost packets of non-reference frames (i.e., B_1, B_2, ..., and B_{10} packets). As expected, PP substantially reduces the number of repair packets required for protecting all reference frames and substantially reduces the maximum number of lost packets of non-reference frames. To examine the effect of permuting data at CM frame level, we repeated this experiment for numerous frame permutation sequences. We



(a) Number of repair packets required by each GOP for guaranteeing that 95% and 99% of all GOPs will not lose any reference frame (i.e., I, P_1, P_2, P_3, and P_4 frames)

(b) 95-percentile and 99-percentile maximum number of lost B packets per GOP

Figure 21. Performance comparison of NPS and PPS for given QoS requirements (buffer size = 3 GOPs, average loss rate = 5%).

found that permuting only frames cannot reshape the probability function of the number of lost packets within each CM frame as PP does, and therefore the results of other sequences tested are similar to the NDS shown in figure 20. In view of this, only the result for normal playback sequence (NPS) is shown here (see Section 2.1 for NPS). Figure 21(a) plots the number of repair packets required by PPS and NPS for guaranteeing that 95% and 99% of the GOPs transmitted will not lose any of the reference frame packets (i.e., I, P_1, P_2, P_3, and P_4 packets); figure 21(b) plots the corresponding maximum number of lost packets of non-reference frames. To better quantify the performance improvement yielded by PP, Table 1 summarizes the reduction achieved by PPS in the number of repair packets required, and

Table 1. The reduction achieved by PPS in the number of FEC repair packets required by each GOP for protecting all reference frames in comparison to NDS and NPS (buffer size = 3 GOPs, average loss rate = 5%).

Average loss length (packets)	PPS vs. NDS (%)		PPS vs. NPS (%)	
	QoS = 95	QoS = 99	QoS = 95	QoS = 99
2	45	45	37	47
3	48	59	52	63
4	59	61	60	61
5	55	61	55	61
6	50	64	48	64
7	50	68	54	64
8	56	64	54	63
9	56	63	54	63
10	53	62	53	62

Table 2. The reduction achieved by PPS in p-percentile maximum number of lost packets of non-reference frames per GOP in comparison to NDS and NPS (buffer size = 3 GOPs, average loss rate = 5%).

Average loss length (packets)	PPS vs. NDS (%)		PPS vs. NPS (%)	
	$p = 95$	$p = 99$	$p = 95$	$p = 99$
2	11	31	11	31
3	33	38	27	38
4	38	39	33	39
5	40	48	36	45
6	40	48	36	43
7	44	48	44	43
8	38	43	38	43
9	41	46	38	44
10	35	52	35	50

Table 2 summarizes the reduction achieved by PPS in the maximum number of lost packets of non-reference frames. In this experiment, the extra cost incurred by PP is the initial delay of 1.5 seconds and the memory space of about 230 KB (three times the maximum GOP size minus the maximum *I* frame size) required by both the sender and receiver. In practice, this extra cost is acceptable for most CM applications [7, 11, 27].

5. Conclusions

In this paper, we considered delivering pre-compressed CM streams over the Internet. We discussed the bursty packet loss behavior on the Internet, and investigated the annoying impacts caused by it on the visual quality of CM streaming. To solve the problems incurred due to bursty packet losses, we have proposed a novel robust end-to-end delivery scheme, termed PP, for CM streaming. PP is designed to be both orthogonal and complementary to conventional error control schemes such as FEC and feedback/retransmission schemes. PP can effectively reshape the probability distribution functions of the number of packet losses within each CM frame so that the probability of losing larger numbers of packets within each CM frame is significantly reduced. The cost of applying PP is a small extra buffer space and a short extra initial delay that are acceptable for most stored CM streaming applications. A series of trace-driven simulations were conducted to validate the effectiveness of PP. Our results showed that in various network conditions, PP effectively reduces the probability of losing larger numbers of packets for each CM frame. If combined with FEC, PP will greatly reduce the number of repair packets required to protect reference frames. If combined with feedback/retransmission-based error control schemes, it will substantially reduce the events in which so many reference frame packets are lost that they cannot be retransmitted on time. From these results, we conclude that PP is a robust transmission technique for CM streaming over the Internet.

References

1. J.C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the Internet," in Proc. ACM SIGCOMM'94, London, UK, 1994, pp. 58–67.
2. J.C. Bolot and A.V. Garcia, "The case for FEC-based error control for packet audio in the Internet," ACM Multimedia Systems, in press.
3. J.C. Bolot and T. Turletti, "Experience with rate control mechanisms for packet video in the Internet," Computer Communication Review, Vol. 28, No. 1, 1998.
4. J.C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for Internet telephony," in Proc. IEEE Infocom'99, 1999, pp. 1453–1460.
5. D. Clark and D. Tenenhouse, "Architectural considerations for a new generation of protocols," in Proc. ACM SIGCOMM'90, 1990, pp. 200–208.
6. D. Comer, Internetworking with TCP/IP, Principles, Protocols, and Architecture, Vol. I. Prentice Hall: Englewood Cliffs, NJ, 1995.
7. J.W. Ding and Y.M. Huang, "An efficient striping strategy for video servers using heterogeneous disk-subsystems," Multimedia Tools and Applications, Vol. 18, No. 2, 2002.
8. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Trans. on Networking, Vol. 14, pp. 397–413, 1993.
9. B. Furth, S.W. Smoliar, and H. Zhang, Video and Image Processing in Multimedia Systems, Kluwer Academic Publishers: Norwell, MA, 1995.
10. H. Ghanbari and V. Seferidis, "Cell loss concealment in ATM video codecs," IEEE Trans. on Circuits Systs. Video Tech., Vol. 3, No. 3, pp. 238–247, 1993.
11. Y.M. Huang, J.W. Ding, and S.L. Tsao, "Constant time permutation: An efficient block allocation strategy for variable bit rate continuous media data," The VLDB Journal, Vol. 8, No. 1, pp. 44–54, 1999.
12. F. Kuo, W. Effelsberg, and J.J. Garcia-Luna-Aceves, Multimedia Communications, Prentice Hall: PTR, 1998.
13. U. Manber, Introduction to Algorithms: A Creative Approach, Addison-Wesley, 1989.
14. A.J. McAuley, "Reliable broadband communication using a burst erasure correcting code," in Proc. ACM SIGCOMM'90, Philadelphia, PA, USA, 1990, pp. 297–306.
15. D.E. McDysan and D.L. Spohn, ATM: Theory and Application, McGraw-Hill, Inc., 1995.
16. C. Metz, "IP QoS: Traveling in first class on the Internet," IEEE Internet Computing, pp. 84–88, Mar. 1999.
17. H.Q. Ngo, S. Varadarajan, and J. Srivastava, "Error spreading: Reducing bursty errors in continuous media streaming," in Proc. IEEE Int. Conf. on Multimedia Computing and Systems, 1999, pp. 314–319.
18. J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," in Proc. ACM SIGCOMM'97, Cannes, France, 1997, pp. 289–300.
19. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical evaluation," in Proc. ACM SIGCOMM '98, Vancouver, CA, 1998.
20. V. Paxson, "End-to-End Internet packet dynamics," in Proc. ACM SIGCOMM'97, Cannes, France, 1997.
21. C. Perkins, "RTP payload format for interleaved media," IETF Audio/Video Transport Working Group, IETF draft: draft-ietf-avt-interleaving-00.txt, Nov. 1998.
22. J. Postel, "The TCP maximum segment size and related topics," RFC 879, Internet Engineering Task Force, Nov. 1983.
23. S. Ramanathan and P.V. Rangan, "Feedback techniques for intra-media continuality and Inter-Media synchronization in distributed multimedia systems," The Computer Journal, Vol. 36, No. 1, pp. 19–33, 1993.
24. L. Rizzo, "Effective erasure codes for reliable computer communication protocols," Computer Communication Review, April 1997.
25. T.L. Sheu and J.T. Fang, "A buffer allocation mechanism for VBR video playback," in Proc. IEEE Int. Conf. on Communication Technology, Beijing, China, 2000, Vol. 2, pp. 1641–1644.
26. B.C. Smith, "Implementation techniques for continuous media systems and applications," Ph.D. Thesis, University of California, Berkeley, 1994.
27. D. Sitaram and A. Dan, Multimedia Servers, Morgan Kaufmann Publishers: San Francisco, CA, 2000.
28. W. Stevens, TCP/IP Illustrated, The Protocols, Vol. 1, Addison-Wesley: Reading, MA, 1994.
29. S.A. Thomas, IPng and the TCP/IP Protocols, Wiley Computer Publishing, 1996.

30. S. Varadarajan, H.Q. Ngo, and J. Srivastava, "An adaptive, perception-driven error spreading scheme in continuous media streaming," in Proc. ICDCS'2000, Taipei, Taiwan, 2000, pp. 475–483.
31. S. Varadarajan, H.Q. Ngo, and J. Srivastava, "Error spreading: A perception-driven approach orthogonal to error handling in continuous media streaming," IEEE/ACM Trans. on Networking, Vol. 10, No. 1, pp. 139–152, 2002.
32. H. Wang and K.G. Shin, "Refined design of random early detection gateways," in Proc. IEEE GLOBECOM'99, Dec. 1999, Vol. 1, pp. 769–775.
33. M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in Proc. IEEE INFOCOM'99, 1999, Vol. 1, pp. 345–352.
34. Q.-F. Zhu et al., "Coding and cell loss recovery in DCT-based packet video," IEEE Trans. Circuits Systems. Video Tech., Vol. 3, No. 3, pp. 248–258, 1993.

Jen-Wen Ding received his B.S., M.S., and Ph.D. degree in Engineering Science from National Cheng Kung University, Taiwan, R.O.C., in 1996, 1998, and 2001, respectively. He received Acer Dragon Thesis Award for his outstanding Master Thesis in 1998. Since 2001, he has been with Kun Shan University of Technology, Taiwan, R.O.C., and is currently an assistant professor of Department of Information Management. His research interests include multimedia storage systems and multimedia communications.

Sheng-Yuan Tseng received his B.S. degree in Computer Science and Information Engineering from Tamkang University, Taiwan, R.O.C., in 1988. He received his M.S. degree in Information Management from National Chiao Tung University, Taiwan, R.O.C., in 1992. He was an associate engineer in Computer Communication Laboratory (CCL) of Industry Technology Research Institute (ITRI), Taiwan, from 1992 to 1993. Since 1993, he has been a lecturer in Department of Information Management of Kun Shan University of Technology, Taiwan, R.O.C. He is currently a Ph.D. student in Department of Engineering Science of National Cheng Kung University, Taiwan, R.O.C. His research interests include multi-objective optimization with genetic algorithms, multimedia communications, and Internet QoS control.

Yueh-Min Huang was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in Engineering Science from the National Cheng-Kung University, Taiwan, R.O.C., in 1982, and both the M.S. and Ph.D. degrees in electrical engineering from the University of Arizona, Tucson, AZ, in 1988 and 1991, respectively. He has been with National Cheng Kung University since 1991, and is currently a professor of Department of Engineering Science. His research interests include distributed multimedia systems, data mining, and real-time systems. Dr. Huang is a member of IEEE Computer Society, the American Association for Artificial Intelligence, and the Chinese Fuzzy Systems Association.