Available online at www.sciencedirect.com
SCIENCE DIRECT®
Real-Time Imaging 11 (2005) 300–309
www.elsevier.com/locate/rti

Robust real-time transmission of scalable multimedia for heterogeneous client bandwidths

Longshe Huo[a,b,*], Wen Gao[a,b], Qingming Huang[a,b]

[a]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
[b]Graduate School, Chinese Academy of Sciences, Beijing 100039, China

Available online 23 June 2005

Abstract

For robust real-time transmission of scalable image and video data over packet-loss networks, a commonly used approach is FEC-based multiple description coding, which protects a scalable bitstream with a fixed number of packets of equal length. In this paper, by considering the problem of applying this approach of multicasting a source to a collection of clients with heterogeneous bandwidths, we propose a novel technique that changes packet length but fixes packet number. In one scenario where different clients access the server via separate links, we study the sensitivity of an optimal solution to the change of packet length, and propose a local search procedure which refines the already computed optimal solutions for other bandwidths. Compared to the scheme that computes an optimal solution individually for each bandwidth, this procedure can achieve comparable performance, however with quite lower time complexity, thus can be used in real-time applications. In another scenario where many clients share a bottleneck link, we present an embedded packetization framework for layered multiple description coding, in which even simple methods with low complexities can achieve good performance. We also propose a local search algorithm to optimize the weighted average performance in case of two layers, and a fast heuristic algorithm which can achieve very good performance tradeoff among all clients in case of more than two layers.
© 2005 Elsevier Ltd. All rights reserved.

1. Introduction

Multiple description coding (MDC) [1] has recently emerged as an attractive framework for robust transmission of image and video data over unreliable channels. Many methods of MDC have been developed over the past decade. One particularly efficient and practical one, called FEC-MDC (FEC-based multiple description coding), is based on priority encoding transmission (PET) [2]. It combines scalable source coding with unequal error protection (UEP) to minimize the impact of lost packets on the reconstruction quality at the receiving ends. The basic idea is to partition a scalable source bitstream into segments of decreasing importance, and protect these segments using progressively weaker forward error correction (FEC) channel codes. This method converts a scalable, prioritized bitstream into multiple non-prioritized descriptions (packets) of equal length, thus can provide graceful performance degradation as packet losses increase, and achieve the best joint economy of source and channel codes. The technique can be applied to many scalable (progressively refinable) image and video coders, such as SPIHT [3], JPEG2000 [4], and 3D SPIHT [5].

Several FEC-MDC algorithms have been proposed [6–10]. These algorithms only try to find the optimal solution for a given transmission rate, and also need to determine the packet number and packet length in advance. Because of these limitations, it is difficult to apply them in real-time multicasting scenarios, where a server needs to send a source to a collection of clients over separate links. As various clients may have different access bandwidths, the server needs to calculate

*Corresponding author. Institute of Computing Technology, Chinese Academy of Sciences, P.O.Box 2704# -31, (JDL), Beijing 100080, China Tel.: +86 10 58858300; fax: +86 10 58858301.
E-mail address: lshuo@jdl.ac.cn (L. Huo).

1077-2014/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rti.2005.05.001

optimal FEC-MDC protection solutions for all possible bandwidths. This, however, requires intensive computation. To alleviate the computational complexity, Stankovic et al. [11] proposed a way to calculate the optimal protection for a base bandwidth and then adjust the protection for other bandwidths by fixing the packet length and using different packet numbers for different bandwidths. In this paper, we argue that when dealing with bandwidth variation, a more suitable solution is to adjust packet length. We first derive a proposition which describes the changing behavior of an optimal protection when the packet length changes, and then present a local search procedure which computes an optimal protection for the first client and only refines it to count for the change of packet lengths for other clients. Compared to the scheme that computes an optimal solution individually for each bandwidth, this procedure can achieve comparable performance, however with quite faster processing speed. This is able to be used in real-time media streaming applications. In the situation where both changing packet length and changing packet number are feasible, the computational complexity of this procedure is also lower than that of Stankovic et al. [11].

When sending a source bitstream to a set of clients partially via a bottleneck link, it is preferable to use layered multiple description coding. Chou et al. [12] proposed a method to split the FEC-MDC framework into two layers, each having the same packet length. The low-bandwidth clients receive only the packets of the base layer, while the high-bandwidth clients additionally receive other packets belonging to the enhancement layer. Since their solution optimizes channel codes only for the low-bandwidth clients, high-bandwidth clients may suffer a significant performance loss. Stankovic et al. [11] modified this method to provide a better tradeoff between distortions seen by both high- and low-bandwidth clients. However, the time complexities of their algorithms are high, and they are also difficult to be extended to more than two layers. In this paper, we propose an embedded packetization framework for layered MDC, in which even simple methods with very low complexities can achieve good performance. In this framework, all layers share the same packets, while each packet is partitioned into multiple parts, each belonging to one layer. We first propose a local search algorithm which can improve the weighted average performance of both high- and low-bandwidth clients, then extend this framework to more than two layers, and present a fast heuristic algorithm by which better performance tradeoff can be achieved for all clients with different access bandwidths.

The remainder of this paper is organized as follows. In Section 2, we briefly review the problem statement of FEC-based multiple description coding (FEC-MDC) technique. In Section 3, we address the problem of multicasting over separate links. In Section 4, we deal with the problem of multicasting over a bottleneck link. In Section 5, we provide our results, and in Section 6 we present our conclusions.

2. FEC-based multiple description coding

The FEC-MDC problem can be stated as a combinatorial optimization problem. In this section, our notations mainly follow that of [10]. Consider the transmission of a scalable bitstream over a packet-loss channel using N packets, each of which has a packet payload length of L bytes. In FEC-MDC framework, as shown in Fig. 1, the source bitstream is divided into L consecutive segments, each of which has $m_i \in \{1, \ldots, N\}$ bytes and is protected by an (N, m_i) systematic Reed–Solomon (RS) code of maximal distance [13], $1 \leqslant i \leqslant L$. The stream of these m_i source bytes followed by the $f_i = N - m_i$ redundancy bytes constitutes the ith channel segment. The jth byte from every channel segment constitutes the jth channel packet, $1 \leqslant j \leqslant N$. If n packets of N are lost, the RS codes ensure that all segments that contain no more than $N-n$ source bytes can be recovered. Since the scalable source bitstream is sequentially refinable, decoding of the ith segment depends on all the previous $i-1$ segments, thus the number of redundant bytes must be monotonically non-increasing in the segment index, i.e., $f_1 \geqslant \ldots \geqslant f_L$. Under this constraint, if at most f_i packets are lost, the receiver can decode at least the first i segments. In this paper, we use an L-dimensional vector $F_L = (f_1, \ldots, f_L)$ to denote an FEC-MDC protection scheme (FPS), where $N-1 \geqslant f_1 \geqslant \ldots \geqslant f_L \geqslant 0$. Let $p_N(n)$ denote the probability of losing exactly n packets out of N and let $c_N(k) = \sum_{n=0}^{k} p_N(n)$, $k = 0, 1, ..., N$, then $c_N(f_i)$ is the probability that the receiver correctly recovers the ith segment. Let $\phi(r)$ be the operational rate-distortion function of the scalable bitstream, which is a monotonically non-increasing function, then the expected distortion of the reconstructed source at the decoder side can be

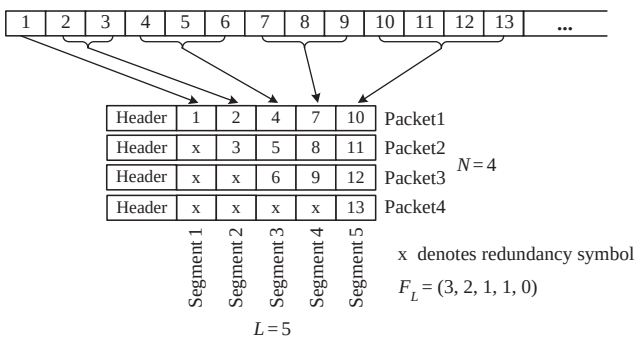

Fig. 1. Example of FEC-MDC packetization.

expressed as

$$E_D(F_L) = c_N(N)\phi(0) - \sum_{i=1}^{L} c_N(f_i)(\phi(r_{i-1}) - \phi(r_i)), \quad (1)$$

where $c_N(N) = 1$, $\phi(0)$ is a constant, $r_i = \sum_{k=1}^{i} m_k = iN - \sum_{k=1}^{i} f_k$, $1 \leqslant i \leqslant L$. The objective of the FEC-MDC problem is to find the FPS $F_L = (f_1, \ldots, f_L)$ that minimizes (1), for given N, L, $\phi(r)$ and $p_N(n)$.

3. Multicasting over separate links

3.1. Change packet number or packet length?

For a given transmission rate, the performance of FEC-MDC depends on the selection of packet number N and packet length L. Increasing N increases the loss recovery capabilities of the RS code, but also increases the redundant header information. The optimal N and L can be determined using exhaustive search method. In certain application situations, the values of N and L can not be chosen arbitrary. For example, when the RS symbols are bytes, the value of N cannot be greater than 255.

For different client access bandwidths, to achieve the best expected reconstruction performance, it is necessary to search the optimal N and L for each transmission rate, and then compute the optimal FPSs for these optimal pairs of N and L. However, this procedure is too complicated and time costly. For simplicity, in practical real-time systems, it is more convenient to fix a parameter (N or L) and only change the other one.

Stankovic et al. [11] chose to fix L and only change N when bandwidth was changed. However, in some situations, this is not the best choice. For example, Fig. 2 shows the optimal N and L for different transmission rates, obtained by encoding the first 16 frames of the sequence *Foreman* using the 3D SPIHT video codec, in the situation where the packet header length is 40 bytes, the maximum packet length is 1460 bytes, the maximum packet number of RS code is 255, and the packet-loss channel is modeled as a two-state Markov process. From the figure we can see that when the transmission rate increases beyond 400 kb, the optimal N is almost fixed as 255, and the optimal L increases nearly linear with the transmission rate. Therefore in this case, for different client access bandwidths, fixing N and only changing L can almost achieve the best FEC-MDC performance. When the transmission rate decreases below 400 kb, changing L by fixing N and changing N by fixing L may result in similar performances. However, when N is changed, the packet-loss channel state $p_N(n)$ should be calculated again for each $0 \leqslant n \leqslant N$, thus the time complexity of changing N is greater than that of changing L. As a result, it is more preferable to choose a fixed N and only change L for different transmission rates.

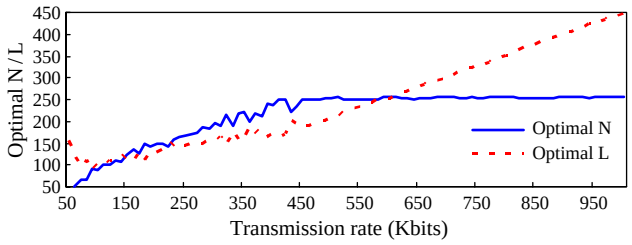

Fig. 2. Optimal packet number (N) and packet length (L) for different transmissions rates.

3.2. Optimal protection changing behavior for changed packet length

For a given scalable bitstream with the operational rate-distortion function $\phi(r)$, packet-loss channel state $p_N(n)$, and fixed packet number N, we introduce a proposition which describes the sensitivity of the optimal FPS to the change of packet length L.

Proposition 1. Let $F_L^* = (f'_1, \ldots, f'_L)$ be the optimal FPS of a scalable bitstream protected with N packets of packet length L, $F_{L+1}^* = (f''_1, \ldots, f''_L, f''_{L+1})$ be the optimal FPS of the same bitstream protected with N packets of packet length $L+1$, and assume the rate-distortion function $\phi(r)$ of this bitstream is monotonically non-increasing and convex, then the protection strength of the first L elements of F_{L+1}^* is stronger than that of F_L^*, i.e., $\sum_{i=1}^{L} f''_i \geq \sum_{i=1}^{L} f'_i$.

The proof of Proposition 1 is presented in Appendix. This proposition can be applied to many scalable source coders, since most of them are constructed to shape an approximately convex operational rate-distortion curve. Although this proposition predicts only the change trend of the total protection strength of the optimal FPS, extensive experiments show that this behavior is also correct for each single segment, i.e., for each $1 \leqslant i \leqslant L$, $f''_i \geqslant f'_i$.

3.3. Refine protections for multiple bandwidths

Based on the above proposition and observation, we present a new local search procedure which first computes an optimal FPS for the lowest bandwidth, and then slightly refines this FPS for higher bandwidths by iteratively increasing the packet length L and the corresponding protection strength. The steps of this procedure are as follows:

Step 1: Sort all bandwidths in ascending order, denoted as $B_1, B_2, \ldots, B_k$.

Step 2: Choose an appropriate packet number N. It should be optimal or near optimal for most of

the bandwidths. For each B_i (in bits), $1 \leqslant i \leqslant K$, the corresponding packet length (in bytes) can be calculated as $L_i = \lfloor B_i/8/N \rfloor - H$, where H is the packet header length.

Step 3: Compute the optimal FPS for B_1 using an existing FEC-MDC algorithm, for example the ones from [7,8]. Let $i = 2$.

Step 4: If $i > K$, stop.

Step 5: Assume the optimal FPS of B_{i-1} is $F_{L_{i-1}} = (f_1, \ldots, f_{L_{i-1}-1}, f_{L_{i-1}})$, then set the current feasible FPS of B_i as $F_{L_i} = (f_1, \ldots, f_{L_{i-1}-1}, f_{L_{i-1}}, f_{L_{i-1}}, \ldots, f_{L_{i-1}})$.

Step 6: Refine the FPS of B_i starting from F_{L_i} using a local search algorithm similar to that of [9].

Step 7: Let $i = i + 1$, go to Step 4.

The local search algorithm of [9] used in Step 6 works by iterative increasing the protection strength. It starts from the current feasible FPS and searches for the best candidate in its neighborhood. If this candidate is better than the current FPS, adopts it as new current FPS and repeats the search from this new FPS. Otherwise, stops. For convenience, we call this local search algorithm *LSU* (local search up), and call its neighborhood *stronger neighborhood*. The definition of *stronger neighborhood* is as follows [9]:

Definition 1. Let $F_L = (f_1, \ldots, f_L)$ be a feasible FPS. The *stronger neighborhood* of F_L consists of the feasible FPSs of the form: $(f_1 + 1, f_2, \ldots, f_{L-1}, f_L)$, $(f_1 + 1, f_2 + 1, \ldots, f_{L-1}, f_L)$, …, $(f_1 + 1, f_2 + 1, \ldots, f_{L-1} + 1, f_L + 1)$.

The above procedure refines protections starting from the lowest bandwidth. It is possible to start from the highest one or a middle one. While refining the optimal FPS for a shorter packet length from a longer one, a similar local search algorithm *LSD* (local search down) should be used. We denote the neighborhood used in *LSD* as *weaker neighborhood*, and its definition is as follows:

Definition 2. Let $F_L = (f_1, \ldots, f_L)$ be a feasible FPS. The *weaker neighborhood* of F_L consists of the feasible FPSs of the form: $(f_1, f_2, \ldots, f_{L-1}, f_L - 1)$, $(f_1, f_2, \ldots, f_{L-1} - 1, f_L - 1)$, …, $(f_1 - 1, f_2 - 1, \ldots, f_{L-1} - 1, f_L - 1)$.

Assume the time complexity of an existing FEC-MDC algorithm is X, then if we compute an optimal solution individually for each bandwidth using this algorithm, the overall time complexity is KX. In our proposed procedure, we only apply the FEC-MDC algorithm once for the lowest bandwidth. In the worst situation, the further refining local search process starts from the first solution $(0, \ldots, 0)$ and stops at $(N-1, f_2, \ldots, f_{L_K})$, this requires $L(N-1)+1$ computations and $L(N-1)$ comparisons of the cost function (1). Hence the overall worst-case complexity of our proposed procedure is $X + O(NL)$. Since the best time complexity of existing FEC-MDC algorithms is $X = O(NL)$ [9], thus our proposed procedure should be at least $K/2$ times faster than the one that computes an optimal solution individually for each bandwidth using an existing FEC-MDC algorithm, where K is the number of all possible client access bandwidths. When K becomes larger, the proposed procedure is beneficial.

4. Multicasting over a bottleneck link

Consider the scenario where a server sends a source to a collection of clients partially through a bottleneck link, with different clients having different access bandwidths. In this case, the server needs to compute only a single protection scheme for the source, and send the protected data via this bottleneck link. Each client receives only parts of the protected data according to its access bandwidth. Since the single protection scheme may not be optimal for all possible access bandwidths, it is necessary to devise an appropriate method which can achieve good quality tradeoff among all clients.

4.1. Layered multiple description coding

Chou et al. [12] addressed this problem by introducing layered multiple description coding (LMDC). For simplicity, they consider only the case of two layers. The base FEC-MDC layer consists of N_1 packets, while the enhancement FEC-MDC layer consists of N_2 packets, with each packet having a fixed length of L bytes. The low-bandwidth clients only receive the base layer, while the high-bandwidth clients receive both the base layer and the enhancement layer. The first N_1 packets are shared by both low- and high-bandwidth codes. For convenience, we call this framework as fixed-length packetization framework (FPF).

There are two simple methods to construct an LMDC in the FPF framework. One is to optimize a single FEC-MDC for high-bandwidth clients, and split it into base and enhancement layers by transmitting only the first N_1 packets to low-bandwidth clients. This may result in a large distortion for low-bandwidth clients. The other method is to optimize a single FEC-MDC for low-bandwidth clients, and transmit all of it, plus N_2 additional parity packets, to high-bandwidth clients. This may cause a large distortion for high-bandwidth clients. In the following we denote these two methods as *FPF-NA* and *FPF-NB*, respectively.

Chou et al. [12] proposed to first optimize the N_1 packets of the MDC base layer for low-bandwidth clients, and then optimize the other N_2 packets in the

MDC enhancement layer to minimize the distortion for high-bandwidth clients. The idea is to borrow some number q of the N_2 packets from the enhancement layer to protect the base layer as additional parity packets, while the other N_2-q packets in the enhancement layer are used to protect the remaining source bytes not already presented in the base layer. This approach (denoted as *FPF-Chou*[12]) can achieve the possible best performance for low-bandwidth clients. However, for high-bandwidth clients, they may still suffer a higher performance loss. The results are also unacceptable for situations where the population of high-bandwidth clients is greater than that of low-bandwidth clients.

To determine the optimal value of q, it is needed to run an FEC-MDC algorithm for at least N_2 times. For larger N_2, the process is time costly, thus the method is also not suitable for real-time applications.

4.2. Embedded packetization framework

In this section, we present a new embedded packetization framework (EPF) for LMDC. In our framework, all clients with different access bandwidths share all the same packets, while each packet is partitioned into several parts, each belongs to one layer. The lowest-bandwidth clients only receive the base layer of each packet. With the increase of bandwidth, more layers are added. The highest-bandwidth clients will receive all layers of each packet. Each packet received by a lower-bandwidth client is always a prefix of the same packet received by a higher-bandwidth client.

To enable the application of EPF, a packet truncating gateway (PTG) is needed in the intermediate node of the network. As shown in Fig. 3, the server forms each packet with the full length, and then sends it via the bottleneck link to the PTG. While receiving a packet, for each client, the PTG truncates only a prefix of the packet and sends to it according to its access bandwidth.

Denote the collection of client access bandwidths as $B_1, B_2, \ldots, B_K$, where $B_1 < B_2 < \ldots < B_K$, and the client number of B_i as C_i, $1 \leqslant i \leqslant K$. Assume the packet number is fixed as N, and each packet has the same header length H, then for each B_i, we can calculate a packet length $L_i = \lfloor B_i/8/N \rfloor - H$ for it. In our EPF framework, for a given layered FEC-MDC construction, if we denote the FPS of B_i as $F_{L_i} = (f_1, \cdots, f_{L_i})$, $1 \leqslant i \leqslant K$, then each F_{L_i} is a prefix of F_{L_K}. Therefore we only need to find an FPS F_{L_K} that can meet the quality demands of all clients jointly.

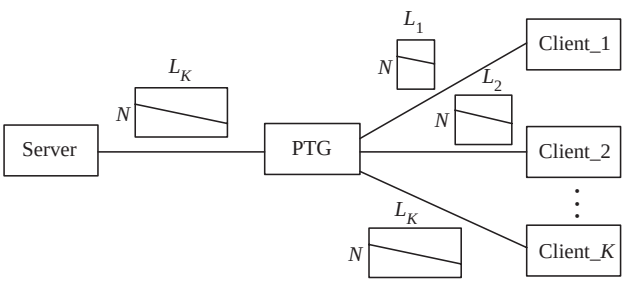

Fig. 3. Embedded packetization framework transmission system.

4.3. Local search by optimizing weighted average performance

For simplicity, we first consider only two layers of EPF, where $K = 2$. We denote the FPS solely optimized for low-bandwidth clients as $F^B_{L_1} = (f^B_1, \ldots, f^B_{L_1})$, and the FPS solely optimized for high-bandwidth clients as $F^H_{L_2} = (f^H_1, \ldots, f^H_{L_1}, f^H_{L_1+1}, \ldots, f^H_{L_2})$. There are also two simple methods we can use to construct an LMDC in the EPF framework. The first one, called *EPF-NA*, is to directly use $F^H_{L_2}$ as the FPS of this LMDC. In this case, high-bandwidth clients can reach their possible minimum distortion, while low-bandwidth clients may suffer some performance losses. The second way, called *EPF-NB*, is to use $F^B_{L_1}$ as a prefix of the LMDC's FPS, and then extends it to the length of L_2 by simply duplicating the last element (denoted as $F^{B'}_{L_2} = (f^B_1, \ldots, f^B_{L_1}, f^{B'}_{L_1+1}, \ldots, f^{B'}_{L_2})$, where $f^{B'}_{L_1+1} = \ldots = f^{B'}_{L_2} = f^B_{L_1}$). This may result in the possible minimum distortion for low-bandwidth clients, and also some performance degradations for high-bandwidth clients. However, experiments show that, compared with the FPF framework, the performance losses of both *EPF-NA* and *EPF-NB* are much lower than that of *FPF-NA* and *FPF-NB*.

If an LMDC is optimized for high-bandwidth clients, it is not fair for low-bandwidth clients especially when their population is greater than that of high-bandwidth clients, and vice versa. In order to achieve a good performance tradeoff between both high- and low-bandwidth clients, we define a weighted average performance measurement, and propose a local search algorithm to optimize it.

Assume the fraction of high-bandwidth clients in total population is h, and the fraction of low-bandwidth clients is $1-h$, $0 \leqslant h \leqslant 1$, then for a given LMDC, its weighted average expected distortion is defined as

$$WE_D(F_{L_2}, h) = hE_D(F_{L_2}) + (1-h)E_D(F_{L_1}), \quad (2)$$

where F_{L_1} is a prefix of F_{L_2}.

The solutions obtained from the two simple methods, $F^H_{L_2}$ and $F^{B'}_{L_2}$, can be seen as two special cases by minimizing (2) with $h = 1$ and $h = 0$, respectively. According to Proposition 1, the protection strength of $F^H_{L_2}$ is stronger than that of $F^{B'}_{L_2}$. If we denote the optimal FPS of minimizing (2) in general case as $F^W_{L_2} = (f^W_1, \cdots, f^W_{L_1}, f^W_{L_1+1}, \cdots, f^W_{L_2})$, we conjecture that the curve of $F^W_{L_2}$ is between that of $F^H_{L_2}$ and $F^{B'}_{L_2}$, i.e., the protection strength of $F^W_{L_2}$ is stronger than that of $F^{B'}_{L_2}$ and weaker than that of $F^H_{L_2}$. Based on this, we devise

two local search algorithms to resolve $F^W_{L_2}$, one starting from $F^{B'}_{L_2}$ and iteratively increasing its protection strength, while the other starting from $F^H_{L_2}$ and iteratively decreasing its protection strength. Due to limitations in space, in the following we only describe the first one.

Define $f_0 = N$ and $f_{L_2+1} = -1$. For $1 \leqslant i \leqslant L_2 + 1$, if $f_i \neq f_{i-1}$, we call i a redundancy change point. In each iterating, we search the neighborhood of current feasible solution from every possible redundancy change point.

Definition 3. Let $F_{L_2} = (f_1, \cdots, f_{L_1}, f_{L_1+1}, \cdots, f_{L_2})$ be a feasible solution for a given LMDC. The neighborhood of F_{L_2} at redundancy change point i, $1 \leqslant i \leqslant L_2$, consists of the solutions of the form: $(f_1, \cdots, f_{i-1}, f_i + 1, f_{i+1}, \cdots, f_{L_2})$, $(f_1, \cdots, f_{i-1}, f_i + 1, f_{i+1} + 1, \cdots, f_{L_2})$, $\cdots$, $(f_1, \cdots, f_{i-1}, f_i + 1, f_{i+1} + 1, \cdots, f_{L_2} + 1)$, which are also feasible solutions for the LMDC.

Algorithm. *EPF-LS*. Local search by optimizing weighted average performance.

Step 1: Compute $F^{B'}_{L_2}$ using an existing FEC-MDC algorithm, for example the ones from [7,8]. Initialize current feasible solution $F_{L_2} = F^{B'}_{L_2}$, set $cont = 1$.

Step 2: If $cont = 0$, stop and return F_{L_2} as the best solution; else set $cont = 0$, $i = 1$.

Step 3: If $f_i = N - 1$, go to Step 6.

Step 4: Search for the solution $F'_{L_2} = (f_1, \cdots, f_{i-1}, f_i + 1, \cdots, f_{i+j} + 1, f_{i+j+1}, \cdots f_{L_2}), 0 \leqslant j \leqslant L_2 - i$, whose weighted average expected distortion is the minimum in the neighborhood of F_{L_2} at redundancy change point i.

Step 5: If $WE_D(F'_{L_2}, h) < WE_D(F_{L_2}, h)$, set $F_{L_2} = F'_{L_2}$ $cont = 1$, and $i = i+j+1$.

Step 6: Repeat $i = i+1$, until i is a redundancy change point. If $i \leqslant L_2$, go to step 4; else go to Step 2.

Compared with *FPF-Chou*[12], the computing speed of *EPF-LS* is faster since it only needs to call once the FEC-MDC algorithm and a local search process whose worst case time complexity is $O(NL)$.

Fig. 4 shows an example of the FPSs resolved by method *EPF-NA*, *EPF-NB*, and the proposed scheme *EPF-LS* with $h = 0.5$. The curve of *EPF-NA* is above that of *EPF-NB*, which means that the protection strength of the FPS resolved by *EPF-NA* is stronger than the one resolved by *EPF-NB*. The curve of *EPF-LS* is situated almost in the middle of the other two curves. This is accordant with our imagination.

4.4. Algorithm optimized for average client bandwidth

When there are more than two bandwidths, i.e., $K > 2$, we extend the definition of weighted average expected distortion as follows:

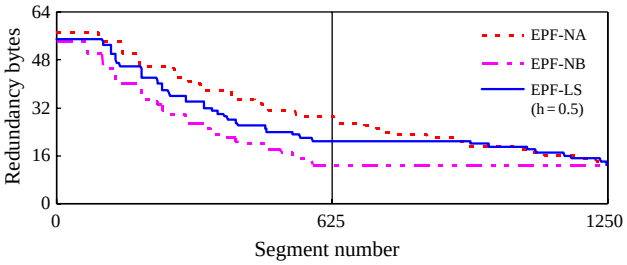

Fig. 4. FEC-MDC protection schemes resolved by *EPF-NA*, *EPF-NB* and the proposed *EPF-LS*.

$$WE_D(F_{L_k}) = \sum_{i=1}^{K} \frac{C_i}{C} E_D(F_{L_i}), \tag{3}$$

where $C = \sum_{i=1}^{K} C_i$, each F_{L_i} is a prefix of F_{L_K}, $1 \leqslant i \leqslant K$. Now the objective is to find an F_{L_K} that minimizes (3).

The *EPF-LS* algorithm is no longer suitable for this situation, because it needs to compute every $E_D(F_{L_i})$ in each of its iterations. When K becomes larger, this is very time costly. Extensive experiments show that in EPF framework, if we use a prefix of the FPS optimized for a longer packet length as an approximate FPS for a shorter one, the expected performance is only slightly lower than the possible best one. It is also true when we use an extended version of the FPS optimized for a shorter packet length as an approximate FPS for a longer one. Based on these observations, we present a simple heuristic algorithm to compute an appropriate solution which can achieve good quality tradeoff among all clients. The idea is to first compute an optimal FPS for the averaged bandwidth of all clients, and then extend it to the length of L_K by simply duplicating the last element. The steps are described as follows:

Algorithm. *EPF-OACB*. Algorithm optimized for average client bandwidth.

Step 1: Let $B_A = \sum_{i=1}^{K} C_i B_i / \sum_{i=1}^{K} C_i$, $L_A = \lfloor B_A/8/N \rfloor - H$.

Step 2: Compute the optimal FPS for B_A using an existing FEC-MDC algorithm, denote it as $F_{L_A} = (f^A_1, f^A_2, \cdots f^A_{L_A-1}, f^A_{L_A})$.

Step 3: Let $F_{L_K} = (f^A_1, f^A_2, \cdots f^A_{L_A-1}, f^A_{L_A}, f^A_{L_A}, \cdots f^A_{L_A})$, return it as the final solution.

The above algorithm only needs to compute an average bandwidth and call once the FEC-MDC algorithm whose best time complexity is $O(NL)$, thus the computational complexity is very low, and it is feasible to be used in online real-time media streaming applications.

5. Results

We test our techniques using the 8 bpp (bits per pixel) gray-scale 512×512 *Lena* image coded with the SPIHT codec [3], and the CIF-format *Foreman* video sequence coded with the 3D SPIHT codec [5]. The first 16 frames of the *Foreman* sequence are encoded as a GOF (Group of Frames). The packet header length is fixed to 40 bytes. All programs run on a PC with Intel Pentium 2.4 GHz processor.

A two-state Markov model is used to simulate the heterogeneous Internet packet loss channel [14]. There are only two states in this model: Good and Bad. In Good state there is no packet loss, while in Bad state the packets are always lost. The model can be fully described by two state transition probabilities. In our experiments, the transition probability from Good to Bad is set to $p_{GB} = 0.01$, and the transition probability from Bad to Good is set to $p_{BG} = 0.09$. The derivation of $p_N(n)$ for this model follows the one described in [15].

5.1. Experiments for multicasting over separate links

In this section, we compare the time complexity and the expected PSNR performance of our proposed technique to that of several other methods.

In our first experiment, the server sends the scalable bitstream of *Lena* to 7 clients via 7 separate links with bandwidths from 100 to 400 Kb. As shown in Table 1, we compare the PSNR of expected distortion and the CPU time by four methods. Firstly, all methods compute an optimal solution for the highest bandwidth using the FEC-MDC algorithm from Mohr et al. [7] (abbreviated as *M*[7]). The first two methods fix $L = 200$ and decrease N, while the last two methods fix $N = 150$ and decrease L. When compute solutions for other bandwidths, the first and the third methods use the *M*[7] algorithm, while the second and the fourth methods use the *LSD* algorithm. The second method corresponds to the one proposed in Stankovic et al. [11], and the fourth method corresponds to the one proposed in Section 3.3 of this paper. The total CPU time used by our proposed local search refining procedure is approximately four times faster than that of the third method.

In the next experiment, the server sends the scalable bitstream of the *Foreman* to 9 clients via 9 separate links with bandwidth from 600 kb to 1 Mb. The method of [11] is no longer suitable for this situation, since for different bandwidths, fixing $N = 255$ and only changing L can always achieve the best performance. As shown in Table 2, we compare the PSNR of expected distortion and the CPU time by two methods. Both methods first compute an optimal solution for the highest bandwidth using the *M* [7] algorithm, and then fix $N = 255$ and compute solutions for other bandwidths by only decreasing L. The first method uses the *M* [7] algorithm, while the second method uses the *LSD* algorithm, which corresponds to the one proposed in Section 3.3 of this

Table 1
Performance (in PSNR of expected distortion) and computational complexity (in CPU time) comparison of four methods, while sending the SPIHT bitstream of *Lena* to seven clients over separate links

Transmission rate (kb)	1.DecN_*M*[7]		2.DecN_*LSD*[11]		3.DecL_*M*[7]		4.DecL_*LSD*(Proposed)	
	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)	PSNR (dB)	Time (s)
400	38.99	0.103	38.99	0.103	38.81	0.060	38.81	0.060
350	38.25	0.079	38.24	0.025	38.13	0.050	38.13	0.001
300	37.35	0.058	37.32	0.009	37.33	0.047	37.33	0.001
250	36.27	0.039	36.20	0.006	36.35	0.042	36.35	<0.001
200	34.97	0.023	34.83	0.003	35.13	0.039	35.12	<0.001
150	33.18	0.013	32.95	0.003	33.38	0.028	33.38	<0.001
100	30.74	0.005	30.41	0.001	30.53	0.018	30.54	<0.001

Table 2
Performance (in PSNR of expected distortion) and computational complexity (in CPU time) comparison of two methods, while sending the 3D SPIHT bitstream of *Forman* to nine clients over separate links

Transmission rate (kb)	1000	950	900	850	800	750	700	650	600
1. DecL_*M*[7]									
PSNR(dB)	36.5	36.25	35.98	35.65	35.39	35.09	34.79	34.44	34.07
Time(s)	0.045	0.041	0.04	0.038	0.038	0.036	0.033	0.031	0.03
2. DecL_*LSD* (Proposed)									
PSNR(dB)	36.5	36.25	35.98	35.67	35.36	35.09	34.79	34.44	34.08
Time(s)	0.044	0.003	0.001	0.003	0.001	<0.001	0.001	0.001	<0.001

paper. The total CPU time used by our proposed local search refining procedure is approximately six times faster than that of the first method.

From the data listed in Tables 1 and 2, three conclusions can be derived: (a) When the transmission rate decreases below 400 kb, with the change of bandwidth, changing L by fixing N and changing N by fixing L may result in similar performances; (b) The local search refining procedure by changing L is faster than the one of changing N [11], partially due to the fact that changing N needs the re-computation of packet-loss channel state $p_N(n)$; (c) Compared with existing FEC-MDC algorithms, for example M [7], our local search refining procedure can achieve comparable PSNR performance, however with substantial lower computational complexity.

5.2. Experiments for multicasting over a bottleneck link

Now we consider the scenario where a server sends the scalable bitstream of *Foreman* to a collection of clients partially via a bottleneck link.

Firstly, we compare the performances of the FPF framework and the EPF framework. For FPF framework, we choose $N_1 = N_2 = 32$ and $L = 1250$ bytes, which is the same as the one used in [12]; for EPF framework, we choose $N = 64$, $L_1 = 625$ bytes, and $L_2 = 1250$ bytes. As shown in Fig. 5, for high-bandwidth clients, both *FPF-NA* and *EPF-NA* can achieve the possible highest PSNR; the *FPF-NB* has a large distortion of nearly 2 dB worse than the best one; *FPF-Chou*[12] improves it, but still has a gap of 0.9 dB worse than the best one; the performance of *EPF-NB* is slightly better than that of *FPF-Chou*[12], however its computational complexity is much lower; the performance of *EPF-LS* with $h = 0.5$ is very close to the best one, and the difference between their PSNRs is only 0.18 dB. For low-bandwidth clients, the possible best PSNR achieved by *FPF-NB* and *FPF-Chou*[12] is less than that of *EPF-NB*, because the parameters they used, N and L, are different; *FPF-NA* has a large distortion of nearly 3 dB worse than that of *FPF-NB*; the PSNRs achieved by *EPF-NA* and *EPF-LS* with $h = 0.5$ are only

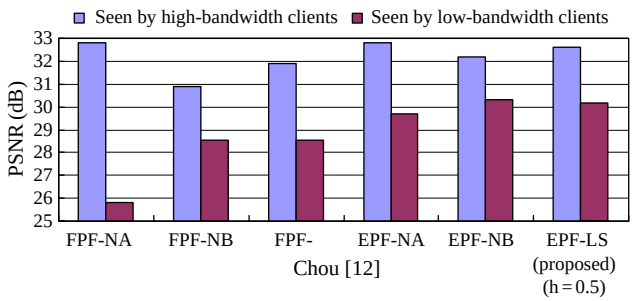

Fig. 5. Performance comparison of different algorithms in the FPF and the proposed EPF framework.

0.6 and 0.15 dB less than the best one achieved by *EPF-NB* respectively. Though the solution of algorithm *EPF-LS* is optimized neither for high-bandwidth clients nor for low-bandwidth clients, its performances for high- and low-bandwidth clients are very close to their possible best ones, suffering no more than 0.2 dB penalties.

Next we analyze the performance of the proposed *EPF-OACB* algorithm. In this experiment, the packet number is fixed as $N = 255$. There are eight possible access bandwidths from 300kb to 1 Mb. We design five client distribution schemes (i.e., the proportion of the number of clients distributed among eight different bandwidths): (1:1:1:1:1:1:1:1), (1:2:3:4:5:6:7:8), (8:7:6:5:4:3:2:1), (1:0:0:0:0:0:0:0) and (0:0:0:0:0:0:0:1). For each scheme, we compute an FPS using the proposed *EPF-OACB* algorithm, and compare the achieved qualities (PSNR of expected distortions) seen by different clients to the possible best ones optimized specially for their bandwidths. Fig. 6(a) shows the comparison of client distribution scheme 1. Fig. 6(b) shows the comparisons of client distribution schemes 2 and 3. Fig. 6(c) shows the comparisons of client distribution schemes 4 and 5. From these we can see that, in each scheme, most of the clients can achieve good expected qualities very near to their possible best upper bounds, while the performance losses of the remaining clients are not very large. Even in the worst situation (the highest bandwidth in scheme 4), the performance loss in PSNR is no more than 1 dB.

Finally, we compare the weighted average performance of the *EPF-OACB* algorithm to that of *EPF_LS* and *FPF-Chou*[12], in the situation that there are only two bandwidths. Fig. 7 shows the PSNR of weighted average expected distortions for different client distribution schemes, achieved by the three algorithms. It can be seen that the performance of *EPF-OACB* is comparable to that of *EPF-LS*, although it has very lower time complexity. These two algorithms can both achieve weighted average expected qualities much better than that of *FPF-Chou*[12].

6. Conclusion

When computing the optimal FEC-MDC protection solutions of a scalable multimedia bitstream for different client access bandwidths, we find that, in most cases, changing packet length is more suitable than changing packet number. We then derive a proposition that describes the change behavior of the optimal protection with the change of packet length. Based on this, we propose a local search refining procedure which can refine the already calculated solutions to be optimal for changed bandwidths with substantial lower time complexity.

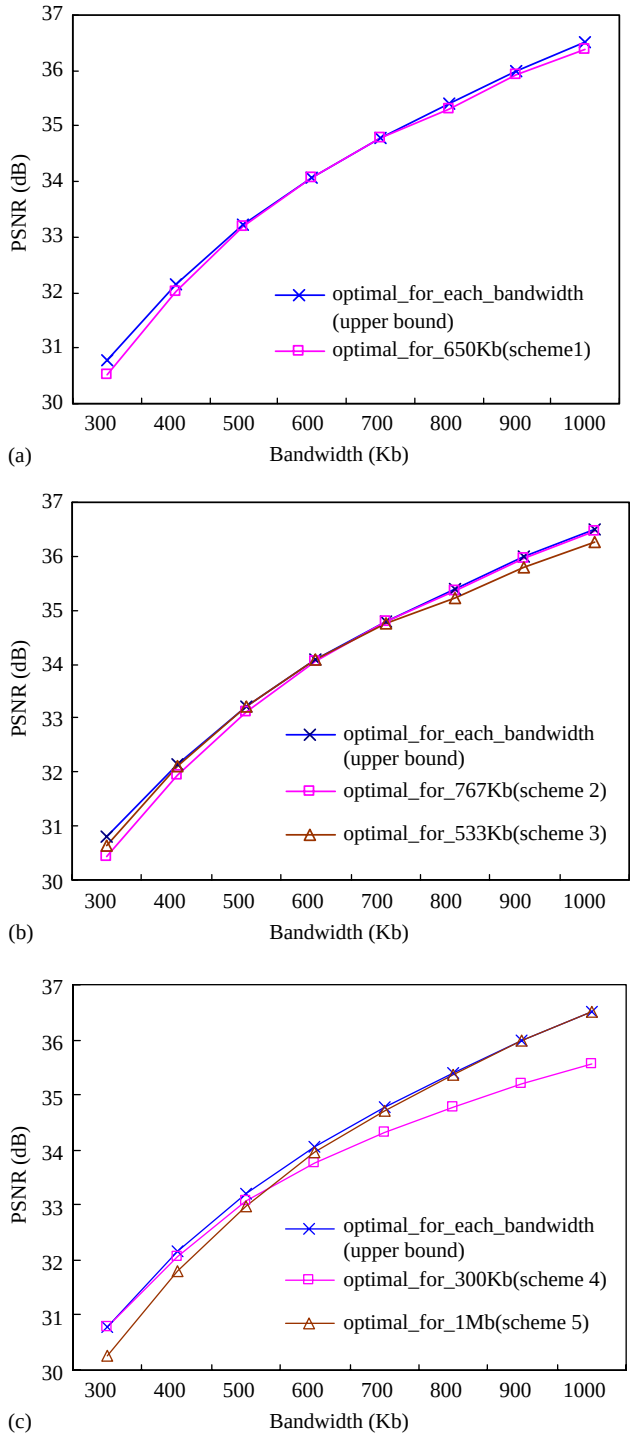

Fig. 6. Performance analysis of the proposed algorithm *EPF_OACB*. PSNR of expected distortion seen by clients with various access bandwidths. (a) for client distribution scheme 1: (1:1:1:1:1:1:1:1); (b) for client distribution scheme 2: (1:2:3:4:5:6:7:8) and client distribution scheme 3: (8:7:6:5:4:3:2:1); (c) for client distribution scheme 4: (1:0:0:0:0:0:0:0) and client distribution scheme 5: (0:0:0:0:0:0:0:1).

In the scenario of multicasting a scalable source to clients with different access bandwidths partially over a bottleneck link, we propose a new embedded packetization framework for layered multiple description coding, in which even simple methods with low complexities can achieve good performance. Besides, we present a local search algorithm and a heuristic algorithm to optimize the weighted average performance for different client distribution schemes. The latter algorithm is quite faster and is able to achieve very good performance tradeoff among all clients in case of more than two layers.

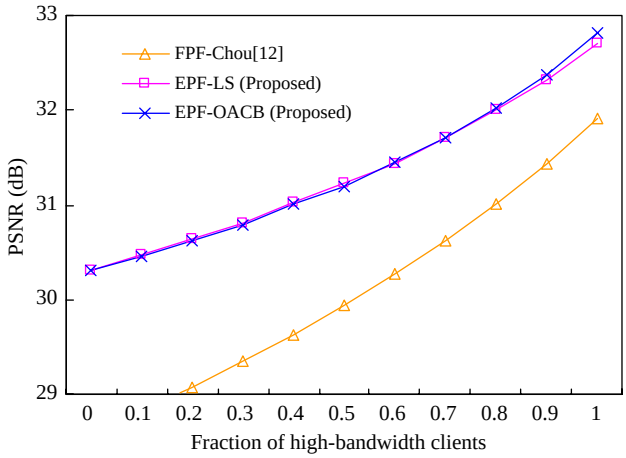

Fig. 7. Performance comparison of three algorithms: *FPF-Chou*[12], *EPF_LS* and *EPF-OACB* for different client distribution schemes, in a situation that there are only two bandwidths.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under grant No. 60333020 and the Natural Science Foundation of Beijing under grant No. 4041003.

Appendix

Proof of Proposition 1. Since the term $c_N(N)\phi(0)$ in (1) is a constant, thus the objective of minimizing (1) is equivalent to that of maximizing

$$E_\Delta(F_L) = \sum_{i=1}^{L} c_N(f_i)(\phi(r_{i-1}) - \phi(r_i)). \tag{4}$$

Therefore, for any $F_L = (f_1, \cdots, f_L)$, there must be

$$\sum_{i=1}^{L} c_N(f_i)(\phi(r_{i-1}) - \phi(r_i)) \leqslant \sum_{i=1}^{L} c_N(f'_i)(\phi(r'_{i-1}) - \phi(r'_i)), \tag{5}$$

where $r'_i = iN - \sum_{j=1}^{i} f'_i$, $1 \leqslant i \leqslant L$; and for any $F_{L+1} = (f_1, \cdots, f_L, f_{L+1})$, there must be

$$\sum_{i=1}^{L+1} c_N(f_i)(\phi(r_{i-1}) - \phi(r_i)) \leqslant \sum_{i=1}^{L+1} c_N(f''_i)(\phi(r''_{i-1}) - \phi(r''_i)), \tag{6}$$

where $r''_i = iN - \sum_{j=1}^{i} f''_i$, $\quad 1 \leqslant i \leqslant L+1$.

We derive the proof by contradiction. Let $t' = \sum_{i=1}^{L} f'_i$, $t'' = \sum_{i=1}^{L} f''_i$, assume $t' > t''$ and $f'_L \geq f''_L \geq f''_{L+1}$, then we can construct a feasible FPS $\bar{F}_{L+1} = (f'_1, \cdots, f'_L, f''_{L+1})$, with

$$E_\Delta(\bar{F}_{L+1}) = \sum_{i=1}^{L} c_N(f'_i)(\phi(r'_{i-1}) - \phi(r'_i)) + c_N(f''_{L+1})(\phi(r'_L) - \phi(r'''_{L+1})), \quad (7)$$

where $r'''_{L+1} = r'_L + (N - f''_{L+1})$. Substituting (7) into (6), we can obtain

$$\sum_{i=1}^{L} c_N(f'_i)(\phi(r'_{i-1}) - \phi(r'_i)) + c_N(f''_{L+1})(\phi(r'_L) - \phi(r'''_{L+1})) \leqslant \sum_{i=1}^{L} c_N(f''_i)(\phi(r''_{i-1}) - \phi(r''_i)) + c_N(f''_{L+1})(\phi(r''_L) - \phi(r''_{L+1})). \quad (8)$$

Exchange term places, (8) can be rewritten as

$$\sum_{i=1}^{L} c_N(f'_i)(\phi(r'_{i-1}) - \phi(r'_i)) - \sum_{i=1}^{L} c_N(f''_i)(\phi(r''_{i-1}) - \phi(r''_i)) \leqslant c_N(f''_{L+1})(\phi(r''_L) - \phi(r''_{L+1})) - c_N(f''_{L+1})(\phi(r'_L) - \phi(r'''_{L+1})). \quad (9)$$

According to (5), the left of (9) is greater than or equal to zero, thus we have

$$c_N(f''_{L+1})(\phi(r''_L) - \phi(r''_{L+1})) - c_N(f''_{L+1})(\phi(r'_L) - \phi(r'''_{L+1})) \geqslant 0. \quad (10)$$

Removing constant term and exchanging place, (10) can be rewritten as

$$\phi(r'_L) - \phi(r'''_{L+1}) \leqslant \phi(r''_L) - \phi(r''_{L+1}), \quad (11)$$

that is,

$$\phi(NL - t') - \phi((NL - t') + (N - f''_{L+1})) \leqslant \phi(NL - t'') - \phi((NL - t'') + (N - f''_{L+1})). \quad (12)$$

Let $A = NL - t', B = NL - t'', C = N - f''_{L+1}$, (12) can be rewritten as

$$\phi(A) - \phi(A + C) \leqslant \phi(B) - \phi(B + C). \quad (13)$$

Since $t' > t''$, $f''_{L+1} \leqslant N - 1$, thus there must be

$$A < B, \quad C > 0. \quad (14)$$

According to (13) and (14), $\phi(r)$ may not be a nonincreasing and convex function, a contradiction.

References

[1] Goyal VK. Multiple description coding: compression meets the network. IEEE Signal Processing Magazine 2001;18(5): 74–93.

[2] Albanese A, Blomer J, Edmonds J, Luby M, Sudan M. Priority encoding transmission. IEEE Transactions on Information Theory 1996;42(6):1737–44.

[3] Said A, Pearlman WA. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Transactions on Circuits and Systems for Video Technology 1996;6(3): 243–50.

[4] Skodras A, Christopoulos C, Ebrahimi T. The JPEG 2000 still image compression standard. IEEE Signal Processing Magazine 2001;18(5):36–58.

[5] Kim BJ, Xiong Z, Pearlman WA. Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT). IEEE Transactions on Circuits and Systems for Video Technology 2000;10(8):1374–87.

[6] Mohr A.E., Riskin E.A., Ladner R.E., Graceful degradation over packet erasure channels through forward error correction. Proceeding of the Data Compression Conference, Snowbird, UT, USA, March 1999.

[7] Mohr A.E., Ladner R.E., Riskin E.A., Approximately optimal assignment for unequal loss protection. Proceeding of the IEEE ICIP-2000, Vancouver, September 2000.

[8] Puri R, Ramchandran K. Multiple description coding using forward error correction codes. Proceeding of the 33rd Asilomar Conference on Signals and Systems, Pacific Grove, CA, October 1999.

[9] Stankovic V, Hamzaoui R, Xiong Z. Packet loss protection of embedded data with fast local search. Proceeding of the IEEE ICIP-2002, Rochester, New York, USA, September 2002.

[10] Dumitrescu S, Wu X, Wang Z. Globally optimal uneven error-protected packetization of scalable code streams. IEEE Transactions on Multimedia 2004;6(2):230–9.

[11] Stankovic V, Xiong Z. Packet erasure protection for multicasting. Proc. Data Compression Conference, Snowbird, UT, USA, March 2004.

[12] Chou PA, Wang HJ, Padmanabhan VN. Layered multiple description coding. Proceeding of the Packet Video Workshop, Nantes, France, April 2003.

[13] Rizzo L. Effective erasure codes for reliable computer communication protocols. ACM Computer Communication Review 1997;27(2):24–36.

[14] Yee JR, Weldon EJ. Evaluation of the performance of error-correcting codes on a Gilbert channel. IEEE Transaction on Communications 1995;43(8):2316–23.

[15] Horn U, Stuhlmüller KW, Link M, Girod B. Robust Internet video transmission based on scalable coding and unequal error protection. Signal Processing: Image Communication 1999; 15(1–2):77–94.