SCALABLE JOINT SOURCE-NETWORK CODING OF VIDEO

By

Yufeng Shan

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Electrical Computer and Systems Engineering

Approved by the
Examining Committee:

Prof. Shivkumar Kalyanaraman, Thesis Adviser

Prof. John W. Woods, Thesis Adviser

Prof. William A. Pearlman, Member

Prof. Biplab Sikdar, Member

Prof. Ivan V. Bajic, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2007

(For Graduation May 2007)

SCALABLE JOINT SOURCE-NETWORK CODING OF VIDEO

By

Yufeng Shan

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Electrical Computer and Systems Engineering

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Prof. Shivkumar Kalyanaraman, Thesis Adviser
Prof. John W. Woods, Thesis Adviser
Prof. William A. Pearlman, Member
Prof. Biplab Sikdar, Member
Prof. Ivan V. Bajic, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2007

(For Graduation May 2007)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to first thank my thesis advisors Prof. Shivkumar Kalyanaraman and Prof. John W. Woods for their constant guidance in my thesis, and valuable advices in my research. I also want to thank Profs. William A. Pearlman, Biplab Sikdar and Ivan V. Bajic for serving on my doctoral committee, and for their helpful suggestions on my thesis.

It's very fortunate for me to work with two distinguished professors as my thesis advisors at the Networks Lab and Center for Image Processing Research (CIPR). I am grateful to the numerous students of both labs, too many to mention individually, who contribute to my research work to various extents.

I acknowledge the financial support I received in my Ph.D program from Army Research Office, Intel Corporation and Rensselaer Polytechnic Institute.

Finally I deeply appreciate my wife Xiu Liu and my son Zhuoheng Shan, who accompany me at US and provided me love, support, and happiness through all the time.

ABSTRACT

Streaming video to diverse users over heterogeneous networks is a challenging problem, and it is critical that video applications work well over wired and multi-hop ad-hoc wireless networks. However, neither the network nor the application can provide these assurances working independently of each other. In this thesis, we look at the joint optimization of the video source coder, channel coder, and network protocols for the robust transmission of video to heterogeneous users simultaneously. We view this problem as a Joint Source-Network Coding (JSNC) problem and systematically investigate a set of schemes to carefully structure this interdependence to maximize gains without limiting future flexibility and evolution. We first investigate a fine grain adaptive forward error correction (FGA-FEC) scheme for encoding and adapting a scalable video bitstream. Our FGA-FEC can encode scalable video in such a way that both the embedded bitstream and the error control codes can be easily and precisely adapted in a multidimensional way to satisfy diverse users without complex transcoding at intermediate nodes. We further generalize this method to work in a multihop wireless network, where product codes and bitstream adaptation are jointly optimized for both packet loss and bit error. To improve the effective throughout of a wireless network, we proposed two link-layer error protection schemes (header CRC and header FEC). A cross-layer two-stage FEC scheme in cooperation with an enhanced MAC protocol (header CRC/FEC) is then proposed to improve performance for multimedia data delivery. The proposed scheme enables the joint optimization of protection strategies across the protocol stack. To efficiently utilize one path of an overlay network, we propose an overlay multi-hop FEC (OM-FEC) scheme that provides FEC encoding/decoding capabilities at intermediate nodes. Based on the network conditions, the end-to-end overlay path is partitioned into segments, and appropriate FEC codes are applied over those segments. Finally, we investigate a distributed FGA-FEC algorithm to work on a congested multihop network, where we do FGA-FEC adaptation whenever permitted and do FGA-FEC decode/recode at edge nodes of congested links.

CHAPTER 1
Introduction

1.1 Motivations

Joint source-channel coding (JSCC) is a well known term in video transmission circles, and refers to the combined optimization of source coding and channel coding based on the statistical characteristics of the underlying channel. Though this approach is more powerful than a pure source-coding approach, it assumes a fairly simple and mathematically elegant model of the "channel". For the future, it will be critical that applications like video streaming work well over wired and multi-hop ad-hoc wireless networks. However, neither the network nor the application can provide these assurances working independent of each other. Here we look at the joint optimization of the video source coder, channel error correction coder and the network protocols for the robust transmission of video to heterogeneous users over challenging network environments considering both packet loss and bit errors inside packets. We view this problem as a Joint Source-Network Coding (JSNC) problem and systematically investigated a specific JSNC approach, called fine grain adaptive FEC (FGA-FEC), to carefully structure this interdependence to maximize the gains without limiting future flexibility and evolution.

Simultaneously streaming video to heterogeneous devices, such as powerful PCs, laptops and handset devices, is a challenging problem, since different users may have different video frame rate/quality/resolution requirements, computational capabilities, and network connections. In order to serve such heterogeneous users, conventional approaches (Windows media, Real player) maintain multiple versions of any piece of media to suit the variety of capabilities and preferences. While streaming, the server sends separate copies of the same bitstream to different users, which is clearly not efficient in terms of network bandwidth utilization as shown in an example in Fig. 1.1, where two users are viewing the same video. The server sends out the same video at the bitrates of 1 Mbps and 256 Kbps, respectively, to different users. The total bandwidth consumed in the bottleneck is 1.25 Mbps.

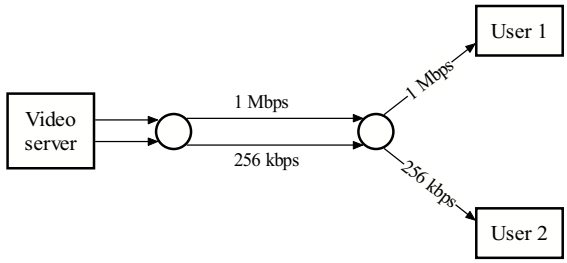


Figure 1.1: Conventional video streaming, same piece of bitstream is sent twice over the bottleneck

IP multicast is an efficient way for simultaneous bulk data delivery. The most serious problem faced by multicast today is the deficiency of its deployment in the wide area network infrastructure. As an alternative, application layer multicast [1] was proposed. In this approach, end systems, instead of routers, are organized into an overlay network to relay data to each other in a peer-to-peer fashion. Application-layer overlay is currently believed to offer us more, since end systems are flexible enough to provide more functions than simple forwarding of packets. Recently, service overlay networks (SON) [2, 46, 47, 8] have gained a lot of attention. Here user defined application level functions are provided at the overlay nodes.

MPEG-21 [9] aims to enable the use of multimedia resources across a wide range of networks and devices. The proposal for the MPEG-21 Part 7 standard is digital item adaptation (DIA), which raises the possibility of in-network video adaptation [9, 10] and fits very well into an overlay infrastructure. Most recently, many papers [10, 42, 43, 44] have reported on frameworks for modeling and arbitrary adaptation of scalable multimedia bitstreams. Scalable video bitstreams have the property that the bitstream corresponding to a low resolution/frame-rate/quality version of the video is embedded in the bitstream corresponding to higher resolution/frame-rate/quality, and can be extracted in a simple manner without transcoding. With the support of a service-overlay network, a single scalable bitstream would be sufficient to satisfy the requirements of both users as shown in Fig. 1.2.

The video server would store one bitstream (corresponding to high frame-rate/resolution/quality) and send it to user 1 at 1 Mbps. The bitstream for user 2 would be extracted from the 1 Mbps stream at an intermediate node and forwarded

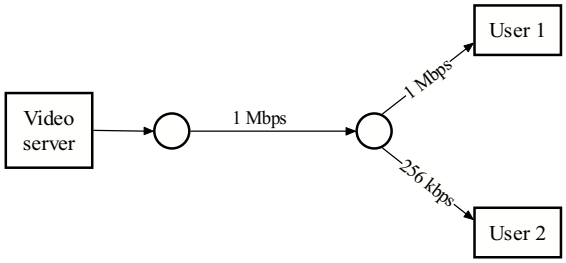


Figure 1.2: Overlay video streaming, low frame-rate/resolution/quality video can be extracted from high frame-rate/resolution/quality video at intermediate overlay nodes

at 256 Kbps. Thus, the total bandwidth consumed in the bottleneck is only 1 Mbps. The potential advantage of using scalable coders becomes even greater in the case of multicast streaming, where different versions of the same content are sent simultaneously to multiple diverse users.

One important issue that is not addressed in MPEG-21 and the recent research is the error-control methods that should be used to match such bitstream adaptation in lossy networks. This thesis addresses the bitstream protection and adaptation problem by a set of joint source-network coding techniques for robust streaming video to heterogeneous users simultaneously.

Our work is different from proxy-based streaming, multicast layered streaming, joint source-channel coding (JSCC) and network coding. Proxy streaming systems cache video content at a local proxy disk and transcode (decode/recode) the video bitstream for different users. In multicast layered streaming, a video server sends different layers to different multicast groups. Receivers adapt to network conditions by joining and leaving these multicast groups, which however, results in a large amount of signaling traffic in a dynamic network. Further, the adaptation is limited to available layers. JSCC is an end-to-end unicast method which jointly optimizes source coding and channel coding, subject to a total transmission rate constraint. Network coding tries to achieve the largest network throughput in a broadcast scenario, by random combining (encoding) packets at certain network nodes. It does not care about the transmitted content and the end users. Our approach only sends one bitstream but can arbitrarily adapt video frame-rate, quality, and spatial resolution without transcoding. Moreover, our approach can provide an efficient

error-control mechanism in cooperation with scalable video to satisfy heterogeneous users simultaneously, thus, extending the scalability features to channel coding.

1.2 Outline of this Thesis

The thesis is organized as follows. Chapter 2 is a literature survey of related research. In Chapter 3, we investigate a fine grain adaptive forward error correction (FGA-FEC) coding scheme for both channel coding and adaptation of scalable video bitstreams. In our work, both the embedded source bitstream and the error-control codes are easily and precisely adapted at intermediate overlay nodes to satisfy multiple heterogeneous users without complex transcoding. We generalize this FGA-FEC algorithm to work with multihop wireless network at Chapter 4, where product codes and bitstream adaptation are jointly optimized for both packet loss and bit errors. To improve effective throughput of a wireless network, we propose two link-layer header-error protection schemes (header CRC and header FEC) in Chapter 5, the intermediate nodes either use header CRC or header FEC to transport packets from source to the destination. In the header error protection scheme, packets with errors in payload are passed to next node or application layer. Therefore, we propose a cross-layer two stage FEC scheme in Chapter 6 to corporate with the header error protection scheme to correct both application layer packet drops and MAC/PHY layer bit errors. The proposed scheme enables the joint optimization of protection strategies across the protocol stack. In both FGA-FEC and its wireless extension schemes, we did not consider the case of network congestion in the backbone. In Chapter 7, we propose an overlay multi-hop forward error correction (OM-FEC) scheme that provides FEC encoding/decoding capabilities at selected intermediate nodes in the overlay path. Based on the network conditions, the end-to-end overlay path is partitioned into segments, and appropriate FEC codes are applied over those segments. In Chapter 8, we investigate a distributed FGA-FEC scheme over a congested multihop network, where we do FGA-FEC decode/recode at selected intermediate overlay nodes, and do FGA-FEC adaptation at other nodes. We propose a coordination method between adjacent nodes to reduce the optimization computation and apply the idea of OM-FEC to reduce the number of FGA-FEC

decode/recode nodes. Finally, we summarize the thesis and suggest possible future work in Chapter 9.

CHAPTER 2
Literature Survey

2.1 Video Coding

Video compression is motivated by the following two facts: (1) raw video contains an immense amount of data, and (2) communication and storage capabilities are limited and expensive. Roughly there are two kinds of video coders: hybrid coders and 3-D transform coders. Hybrid coders are the basis of all current video coding standards. Subband/wavelet transform coders are more popular in the research community. The most recent JPEG2000 [11] image compression standard is based on 2-D subband/wavelet transform.

2.1.1 Hybrid Video Coding

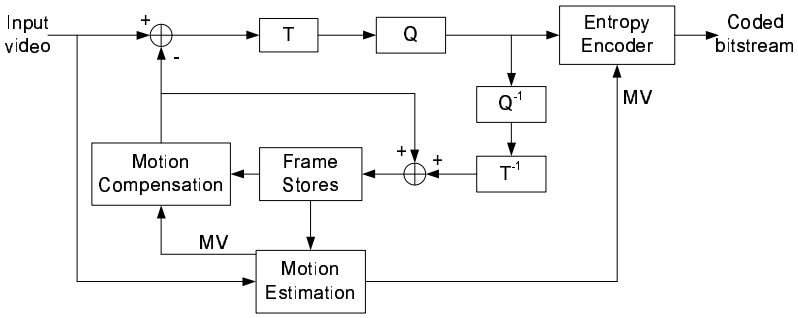
Table. 2.1 lists the most popular video coding standards. They are all based on hybrid video coders.

Table 2.1: Current video coding standards

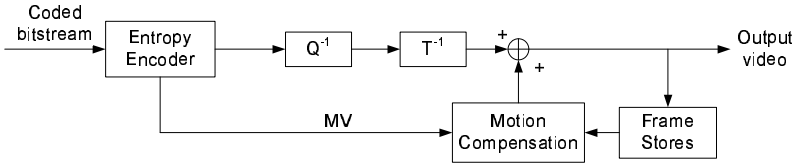
Video coding standard	Primary intended applications	Bitrate
MPEG-1 [12]	Video CD, Audio (MP3)	1.5 Mbps
MPEG-2 [13]	Digital TV, DVD	2-20 Mbps
MPEG-4 [14]	Interactive, mobile services and video streaming	Variable
H.261 [15]	Video conferencing over ISDN	nx64 Kbps
H.263 [16]	Video telephone over PSNT	33.6 Kbps and up
H.264/AVC [17]	Improved video compression	Variable

Due to the high similarity between adjacent video frames, and high correlation among nearby pixels within the same frame, efficient video compression significantly relies on effective removal of temporal and spatial redundancy in the input video. In a conventional video coding system, such temporal and spatial redundancy is exploited in the so called *hybrid coding* technique, a hybrid of motion compensated

(MC) temporal differential pulse code modulation (DPCM) and spatial transform coding.



(a) Encoder



(b) Decoder

Figure 2.1: Block diagram of a conventional hybrid video coding system

The basic block diagram of hybrid coding is shown in Fig. 2.1. During encoding, a sequence of video frames is first divided into GOPs (Group of Pictures) and then input to the hybrid video coder (Fig. 2.1(a)). The first frame of the GOP is usually coded independently of other frames, and is called an I frame. Other frames of the GOP are coded through prediction: a P (predicted) frame is encoded from previously coded I or P frame, and a B (bi-directional predicted) frame is encoded from the previous and following P frames. To encode a P frame from a previous I frame, the P frame is first divided into *macroblock*s (typically a 16×16 block of pixels). The encoder then tries to find the matching block in previous I frame for each microblock, and a vector, called the motion vector (MV), is calculated to connect the two macroblocks. The corresponding block of pixels in the I frame are subtracted from the P frame pixels and the prediction residual is quantized and coded. After motion compensation, the frames are subject to spatial coding using the 2-D discrete cosine transform (DCT) or an approximation represented by the T block in Fig. 2.1(a), where each macroblock is divided into four 8×8 blocks in MPEG-2 and each such block is transformed by DCT. The resulting coefficients are

then quantized by the Q block, and encoded by either Huffman [18] or arithmetic [19] entropy coder. The block Q^{-1} simple maps quantizer indices back to output values.

The decoding process is similar as encoding, but in a reverse order. The video bitstream is first decoded by an entropy decoder and then passed to inverse quantization and inverse transformation, finally a reconstructed video sequence is produced.

2.1.2 Subband/Wavelet Video Coding

Like the DCT, the discrete wavelet transform mathematically transforms an image into frequency components. The process is performed on the entire image, which differs from the DCT method, that works on small blocks of the desired data. The result is a hierarchical representation of an image, where each layer represents a spatial frequency subband. Encouraged by some attractive properties in subband/wavelet image coding, such as in embedded zerotree wavelet (EZW) [32], JPEG2000 [11] and SPIHT [33], three-dimensional (3-D) transform coding is an efficient alternative for video compression. Fig. 2.2 provides a block diagram of a general 3-D subband/wavelet coding system. In comparison to Fig. 2.1, this transform based system does not contain the closed DPCM loop in conventional hybrid coders.

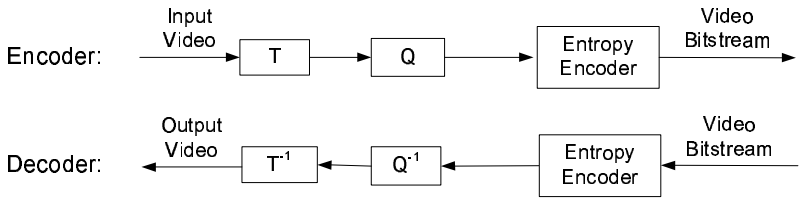
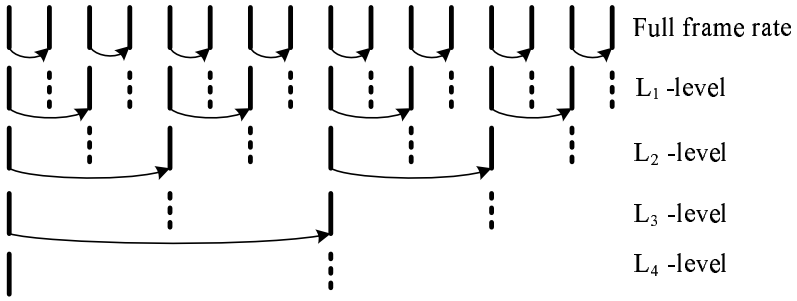


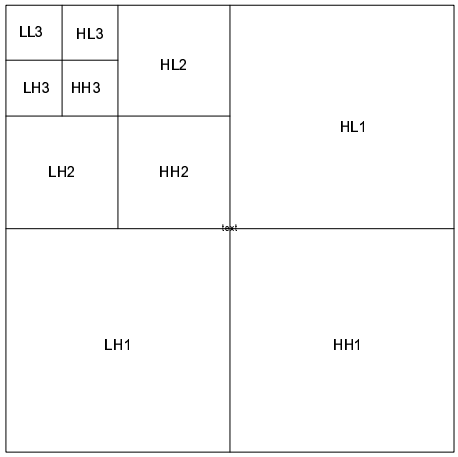
Figure 2.2: Block diagram of the 3-D transform coding system

The common 2-D separable filter bank for 3-D subband analysis/synthesis was first generalized by Karlsson and Vetterli [20], where a pair of 2-tap Haar filters associated with frame average and difference were adopted for temporal filtering. Spatiotemporal or 3-D transform coding with motion compensation was pioneered by Ohm [21]. In his paper, the 3-D subband filter bank is successfully combined

with the block matching motion estimation algorithm, and filtering is performed along a determined motion trajectory. This 3-D MC subband framework is later further developed in [22] and [23]. In the case of MC prediction, the filters are in principle LPC analysis and synthesis filters, while in cases of transform or wavelet coding, transform basis functions are subject to MC alignment. This is denoted as motion-compensated temporal filtering (MCTF). Lifting scheme is introduced into sub-pixel MCTF to realize perfect reconstruction [24, 25, 26]. Currently there are several video coders that use subband/wavelet coding, such as 3-D SPIHT [27], MC-EZBC [28], LZC [31], etc. The basic approach to subband/wavelet video coding is illustrated in Fig. 2.3.



(a) Temporal decomposition



(b) 3-stage spatial decomposition

Figure 2.3: Temporal and Spatial decomposition of a 16-frame GOP

Fig. 2.3(a) shows a typical GOP structure with 16 video frames. The top level represents the video at full frame rate. These incoming frames are input to motion estimation and the resulting motion vectors (indicated by curved arrows) are used

for motion-compensated temporal filtering (MCTF). In this example, neighboring frames are decomposed using a MC Haar filter bank to produce the temporal low frequency bands (solid lines) and temporal high frequency bands (dashed lines) at the next lower temporal level. This process is repeated until we obtain the MC average of all 16 frames in the GOP, at the bottom of the temporal pyramid shown in Fig. 2.3(a). Video data in this example has five temporal scalability layers, going from full frame rate down to L4-level at 1/16 of full frame rate. Only the lowest temporal frequency frame, but all the high temporal frequency frames (dashed lines) are coded, since these frames are necessary and sufficient to reconstruct the original GOP.

The frames which are to be encoded are also subject to 2-D spatial subband/wavelet analysis as shown in an example of 3-stage spatial decomposition in Fig. 2.3(b). In the first stage, frames are low- and high-pass filtered and subsampled, both horizontally and vertically, producing four 1^{st} level subbands: LL1, HL1, LH1 and HH1. Subbands are labeled by the type of horizontal filtering, the type of vertical filtering, and their stage, for example, HL1 stands for high-pass (H) horizontal filtering and low-pass (L) vertical filtering in the first stage. The second stage of subband filtering is performed on the LL1 subband, replacing it by four 2^{nd} level subbands: LL2, HL2, LH2 and HH2. The third stage of decomposition would be performed on LL2, replacing it by four 3^{rd} level subbands LL3, HL3, LH3 and HH3. After spatial decomposition, the subbands can be encoded using a variety of techniques, such as EZBC, EBCOT, and SPIHT. Here, we briefly overview two coders, EZBC and SPIHT, which were developed at RPI.

2.1.2.1 EZBC Image Coding

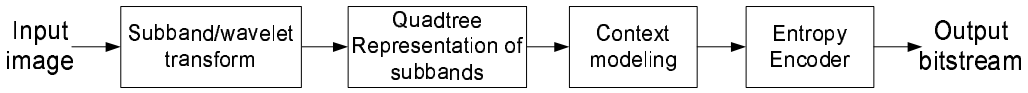


Figure 2.4: Block diagram EZBC image coding

The block diagram of the EZBC (Embedded ZeroBlock Coding and context modeling) image coding algorithm [29] is shown in Fig. 2.4. It includes the following

four blocks:

- subband/wavelet transformation of the input image,
- quadtree representation of subband samples,
- context modeling, and
- entropy coding.

We now briefly describe each of these steps. The first step of the process is the dyadic subband/wavelet transformation of the input image. The transformation is carried out using the Daubechies 9/7 filterbank [30], also used in the recent JPEG2000 image coding standard [11]. If L is the number of levels of the subband/wavelet decomposition, $3L+1$ subbands are obtained. They are indexed 0, 1, ..., $3L$, starting from the lowest frequency subband.

After transformation, a quadtree representation of samples is built up for each subband. Let k be the subband index and $c_k(i,j)$ be the value of the subband coefficient at position (i,j) in subband k. The value of a quadtree node $Q_k[l](i,j)$ at level l is defined recursively as [29]:

$$\begin{aligned} Q_k[0](i,j) &\triangleq |c_k(i,j)|, \\ Q_k[l](i,j) &\triangleq \max\{Q_k[l-1](2i,2j), Q_k[l-1](2i,2j+1), \\ &\qquad Q_k[l-1](2i+1,2j), Q_k[l-1](2i+1,2j+1)\}. \end{aligned} \tag{2.1}$$

In this notation, node $Q_k[l](i,j)$ at level $l > 0$ has four child nodes at level $l-1$: $Q_k[l-1](2i,2j)$, $Q_k[l-1](2i,2j+1)$, $Q_k[l-1](2i+1,2j)$, and $Q_k[l-1](2i+1,2j+1)$. Node $Q_k[l](i,j)$ is called the parent node of its child nodes. The values of the quadtree nodes at level 0 are defined to be the magnitudes of the corresponding individual coefficients. The values of the nodes at the next higher level are set to the maximum value of the corresponding four child nodes in the current quadtree level. In this way we arrive at the following representation of a subband. Level 0 of the quadtree contains the magnitudes of all coefficients of the subband; level 1 contains the largest magnitudes in 2×2 coefficient blocks; level 2 contains the

largest magnitudes in 4×4 coefficient blocks; and so on. The highest level of the quadtree contains the largest magnitude of all coefficients in the subband.

After quadtree construction, subband coefficients are progressively encoded bitplane-by-bitplane, starting from the most significant bit (MSB) towards the least significant bit (LSB). The index of the most significant bitplane is given by $n_{\max} = \llcorner \log_2(\max_{i,j,k} |c_k(i,j)|) \lrcorner$. The magnitude of any coefficient can be written in binary expansion as

$$|c_k(i,j)| = b_0^{(i,j,k)} 2^{n_{\max}} + b_1^{(i,j,k)} 2^{n_{\max}-1} + b_2^{(i,j,k)} 2^{n_{\max}-2} + \cdots , \tag{2.2}$$

where $b_m^{(i,j,k)} \in \{0,1\}$ represents the value of the bit in the bitplane $n_{\max} - m$. Encoding of biplane n ($n = n_{\max}, n_{\max-1}, \cdots$) is done independently for each l in two stages (passes): significance pass and refinement pass. In the significance pass for bitplane n, each quadtree node $Q_k[l](i,j)$ is compared to the threshold $\tau_n = 2^n$, and is considered to be significant if $Q_k[l](i,j) \geq \tau_n$, otherwise it is insignificant. The result of the significance test is represented as a binary symbol: 1 for significant, 0 for insignificant. If a node is found to be insignificant with respect to τ_n, all its descendants must also be insignificant. If a node is found to be significant, its children are also tested for significance. The process is recursively repeated until quadtree level 0. After significance testing for bitplane n is completed, the refinement pass is executed. In this pass, all coefficients which have previously (i.e. in bitplanes $n+1$, $n+2$,$\cdots$, $n_{\max}$) been found to be significant are refined. That is, the value of their bit in bitplane n (i.e. $b_{n_{\max}-n}^{(i,j,k)}$ in (2.2)) is encoded.

The final bitstream is produced by context-based entropy coding of the results of significance tests, sign predictions and refinement bits. This kind of context modeling efficiently exploits both inter- and intra-band dependencies among subband coefficients. Sign predictions and refinement bits are coded using contexts derived from spatially neighboring coefficients. Interested readers can find the details of EZBC in [29].

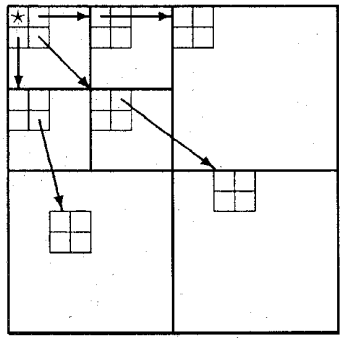


Figure 2.5: Examples of parent-offspring dependencies in the spatial-orientation trees

2.1.2.2 SPIHT

SPIHT [33] was proposed by Amir Said and Willian Pearlman and is computationally very fast and among the best image compression algorithms known today. The "set partitioning in trees" working on each bitplane is actually doing the role of the entropy coder, which gives very little loss of coding efficiency even without using popular entropy coding, such as adaptive arithmetic coding.

In order to reduce the number of decisions in bit comparisons, the set partitioning rule is defined using an expected ordering in the hierarchy implied by the subband pyramid. The natural objective here is to derive new partitions such that those expected to be insignificant contain a large number of elements (i.e. bits in a zerotree), and others expected to be significant contain only one element.

A tree structure, called *spatial orientation tree*, naturally defines spatial relationships on the hierarchical pyramid. Fig. 2.5 (adopted from [33]) shows how the spatial orientation tree is defined in a pyramid constructed with recursive four-subband splitting. Each node of the tree corresponds to a pixel and is identified by the pixel coordinate. Its direct descendants (children) correspond to the pixels of the same spatial orientation in the next finer level of the pyramid. The tree is defined in such a way that each node has either no children (leaf nodes) or four children, which always form a group of 2×2 adjacent pixels. In Fig. 2.5, the arrows are oriented from the parent node to its four children. The pixels in the highest level of the pyramid are the tree roots and are also grouped in 2×2 adjacent pixels.

The encoding procedure and details can be found in the original paper [33].

2.2 Scalability

In the past decade, scalable compression techniques have been widely explored. Scalable video coding schemes are intended to encode the signal once at highest resolution, but enable decoding from partial streams depending on the specific rate and resolution required by a certain application. An important feature of a scalable bitstream is that a portion of the bitstream can be discarded without compromising the usefulness of the more important portions. This enables a simple and flexible solution for adaptation to the variety of channels, terminals, and storage devices. Here, we briefly describe how to achieve scalability using both hybrid DCT and transform-based subband/wavelet coders. We only consider three basic types of video scalability: spatial scalability (resolution), temporal scalability (frame-rate) and quality (SNR) scalability. Both hybrid-based scalability and subband/wavelet-based scalability have been active areas of research, which had led to MPEG-4 fine granularity scalability (FGS) [34] (hybrid coder) and 3-D SPIHT [27], MC-EZBC [28], and LZC [31] in subband/wavelet scalable coding.

2.2.1 SNR Scalability

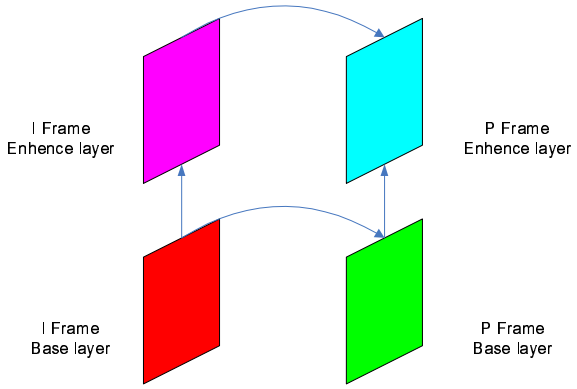


Figure 2.6: SNR scalability of hybrid coder

The typical architecture to produce a scalable bitstream in a hybrid coder is as follows. The DCT-transformed video frame is first quantized by a coarse quantizer, then dequantized and subtracted from the non-quantized frame, and the difference

is then quantized by a finer quantizer [35]. This procedure can be performed several times to produce layers of SNR scalability. Fig 2.6 sketches how to refine the amplitude resolution of an I frame and a P frame, where the base layer uses a coarse quantizer, and the enhancement layer applies a finer quantizer to the difference between the original DCT coefficients and the coarsely quantized base-layer coefficients.

The subband/wavelet coders naturally produce bitstreams which are highly SNR scalable. Lower quality video bitstreams are embedded in the higher quality video bitstreams. This is achieved by the use of embedded quantizers where quantization cells of coarse quantizers are the union of quantization cells of finer quantizers. With the bitstreams produced by these coders, each decoded bit refines the quality of the signal reconstructed from the previously decoded bits.

2.2.2 Temporal Scalability

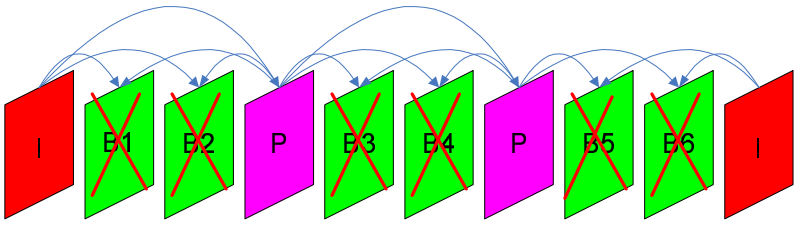


Figure 2.7: Frame-rate scalability of hybrid coder

In a hybrid coder, temporal scalability is obtained by using B frames, which are dependent on I and P frames on either side, however no other frames depend on B frames. Thus, each B frame may be discarded without affecting other frames. Fig. 2.7 sketches the dependency of frames in a nine-frame GOP. Decoding only the I frames gives us the lowest frame rate video, called the *base layer*. Other frames would construct enhancement layers. One can remove B1, B3, B5, or B2, B4, B6, or the two P frames progressively, to obtain lower frame rates.

For a subband/wavelet coder, frame-rate scalability is achieved by decoding the corresponding subbands of the bitstream. For example, decoding the video at 1/2 frame rate corresponds to decoding all temporal subbands up to L_1 level at Fig. 2.3(a). Decoding the subbands to the L_2-level would result in a video at 1/4 frame

rate, and so on. The number of temporal scalability levels is bounded by the size of the GOP, usually, i=$\log_2(\|GOP\|)$, where $\|GOP\|$ is the number of frames in one GOP.

2.2.3 Spatial Scalability

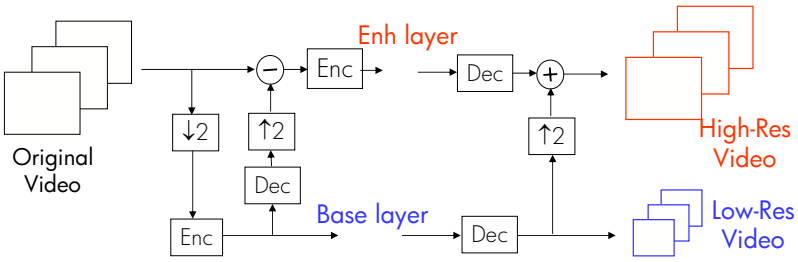


Figure 2.8: Spatial scalability of hybrid coder

In a hybrid coder, resolution scalability can be achieved by first downsampling the original sequence, coding the low resolution sequence and then coding the error between the interpolated low resolution sequence and the original. Fig. 2.8 illustrates two layers of resolution scalability. The base layer is obtained by lowpass filtering and downsampling the original bitstream, while the interpolation residual is called the enhancement layer. With both base layer and enhancement layer, one can reconstruct the original sequence with the full resolution. One can produce more than two layers of resolution scalability, by performing the filtering and downsampling several times and then coding the residual for each layer.

In a subband/wavelet coder, similar to temporal scalability, resolution scalability is also achieved by decoding corresponding subbands of the bitstream. For example, decoding the video at 1/4 resolution corresponds to decoding all spatial subbands except the subbands of LH1, HL1, and HH1 at Fig. 2.3(b). To achieve 1/16 spatial resolution, one should further remove the subbands of LH2, HL2, and HH2 from the bitstream. The number of levels of the spatial SWT determines the number of resolution scalability layers.

2.3 Protection of Scalable Bitstream

In many aspects, transmission of video is the most challenging problem in multimedia communication, due to its real-time nature and large bandwidth requirement, coupled with the lack of quality of service (QoS) guarantees in today's Internet and wireless networks. In the future, motion pictures will mostly be transmitted over variable bandwidth channels, both in wireless and wired networks. They have to be stored on media of different capacity, and be replayed on a variety of devices, ranging from small mobile terminals to high-resolution projection systems. Due to the lossy characteristic of all kind of channels, protection of scalably encoded video bitstream is necessary. Much research have been done in the past decade to protect progressive (embedded) encoded data from losses. One particularly efficient and practical method is based on the Priority Encoding Transmission (PET) technique proposed by Albanese *et al* [36]. PET is a packetization scheme that combines layered source coding with unequal erasure protection. Albanese *et al* applied the PET scheme to the I, P, and B layers of MPEG video, but did not optimize the code rate to minimize the end-to-end distortion for a given overall transmission.

Several algorithms have been proposed for optimal FEC assignment for progressive data based on PET. Mohr *et al* [37] described an algorithm to achieve an approximately optimal assignment of forward error correction to progressive data by a local search algorithm, essentially a Lagrangian optimization. The algorithm first finds an optimal assignment under convex hull and fractional bit allocation assumptions, and then relaxes those constraints to find an assignment that approximates the global optimum. Stankovic *et al* [38] presented an efficient iterative improvement algorithm for greedy search from a near optimal initial condition. The proposed scheme is faster than that of [37]. Puri and Ramchandran [39] provided a Lagrange multiplier based algorithm, which also starts by computing the h vertices of the convex hull of p points of the operational distortion-rate ($D(R)$) function. Then, after an $O(h)$ step, it requires several Lagrange iterations to find the optimal assignment. Stockhammer and Buchner [40] presented a dynamic programming algorithm that is close to optimal in the general case, and optimal if the operational $D(R)$ function is convex and the packet loss probability is a monotonically

decreasing function of the number of packets. Dumitrescu *et al* [41] proposed an approach based on a global search, which finds a globally optimal solution for both convex and non-convex utility length characteristics with similar computation complexity. To protect progressive data in a lossy wireless channel, Sherwood and Zeger [121] proposed a source-channel coding system where the source code is an embedded bitstream and the channel code is a product code such that each row code is a concatenation of a cyclic redundancy check (CRC) and a rate-compatible punctured convolutional code (RCPC). The column codes are RS codes. Stankovic *et al* [118, 119] improved this system [121] for wireless applications by efficiently reorganizing the source code into a set of independently decodable packets, making it more robust in time-varying channels. Cho and Pearlman [120] presented a multilayered protection of embedded video for both bit errors and packet loss using error resiliency and error concealment in a 3-D SPIHT coder. For higher protection against channel noise, the authors used a product code. These steps provide the robust source coder with additional layers of protection against channel noise.

2.4 Bitstream Adaptation and Networking

The proposal for the MPEG-21 [9] Part 7 standard is Digital Item Adaptation (DIA), which raises the possibility of in-network video adaptation [9, 10]. To adapt a multimedia bitstream for multiple users, Mukherjee *et al* [42, 43] developed a metadata-based method called structure scalable meta-format (SSM) [44], that enables flexible yet fully format-agnostic adaptation of a wide class of scalable bitstreams, in a variety of delivery architectures. To transmit data to multiple users, IP multicast and application-layer multicast [1] were proposed for simultaneous bulk data delivery. In the past several years, SONs [45, 46, 47, 8] are gaining attention, in which user-defined application-level functionalities are provided at overlay nodes, other than simple forwarding of packets. The SON fits very well into the MPEG-21 digital item adaptation infrastructure and opens the possibility of practical utilization of DIA techniques to serve heterogeneous users. Most recently, a very hot area in networking research is network coding [56]. The core notion of network coding is to allow and encourage mixing of data at intermediate network nodes. The major

finding in [56] is that, contrary to one's intuition, it is in general not optimal to consider the information to be multicast in a network. Rather, network coding has to be employed to achieve optimality. Li *et al* at [57] proposed an easy and fast linear network coding scheme, and showed that the code is sufficient to achieve the optimal network throughput. A distributed randomized network coding approach is proposed by Ho *et al* [58, 59] for transmission and compression of information in general multi-source multicast networks. Network nodes independently and randomly select linear mappings from inputs onto output links over some field. Chou *et al* [60] extended the randomized network coding idea and showed that it is practical in real world networks. The coding information is carried in the packet header, and the receiver only needs to process the header to decode the information. Microsoft has recently announced a prototype called Avalanche [63] for large scale content distribution on peer-to-peer networks that uses network coding as its core technology. The simulation results show that Avalanche can improve the download time over BitTorrent [64] by 20 to 30%. Essentially, network coding is one good example of using SON.

Overall, the development of currently scalable coding techniques, error-control coding methods and overlay networking opens an exciting research direction for transmission of video to multiple diverse users simultaneously.

CHAPTER 3
Scalable Video Streaming with Fine Grain Adaptive Forward Error Correction

In this chapter, we investigate a fine grain adaptive forward error correction (FGA-FEC) coding scheme for both channel coding and adaptation of scalable video bitstreams. In our work, both the embedded source bitstream and the error-control codes are easily and precisely adapted at intermediate overlay nodes to satisfy multiple heterogeneous users without complex transcoding. The proposed FGA-FEC scheme encodes and adapts the scalable bitstream in such a way that if part of the video source data is actively dropped, parity bits protecting that piece of data are also removed, yielding an efficient result without FEC transcoding. Encoding once at the source, the new method can satisfy multiple heterogeneous users simultaneously without decoding/recoding FEC at intermediate network nodes.

3.1 Introduction

Simultaneously streaming video to heterogeneous devices, such as powerful PCs, laptops, and handset devices, is a challenging problem, since different users may have different video frame-rate, resolution, and quality preferences, or computational and connection-link capabilities. As we mentioned in previous chapters, most recent research either focus on adapting a multimedia bitstream for multiple users without applying error control methods [10, 106, 42, 43], or focus on developing end-to-end optimization schemes to protect a progressive bitstream without any adaptation being considered for diverse users [36, 39, 37, 38]. Chou *et al* [60] presented and evaluated constructions for two-layer multiple description codes using FEC to satisfy various user preferences. Stankovic *et al* [107] modified the method of [60] and defined an optimal layered multiple-description code as one that minimizes the largest performance loss experienced by any client. Both of the papers focus on quality adaptation and do not consider adaptation for diverse users with different frame rate and/or resolution needs.

This study explores the feasibility of using a service overlay network to address the problem of streaming video to heterogeneous users simultaneously. The challenge is to encode the video to facilitate efficient and precise adaptation of the encoded bitstream (adapting both the video bitstream and the error-control codes) to satisfy multiple users without complex transcoding at intermediate overlay nodes. Here, by 'adapt' we mean reducing one or more of the three scalability dimensions (frame-rate, resolution, and quality or source bitrate) of a coded video bitstream along with the corresponding error-control codewords.

The main contribution of this study is a Fine Grain Adaptive FEC scheme which enables multidimensional, arbitrary, and efficient, yet near optimal adaptation of both the encoded video bitstream and the error-control codes to serve multiple diverse users. Our work is different from multicast layered streaming and proxy-based streaming. In multicast layered video streaming, a server sends different layers to different multicast groups. Receivers adapt to network conditions by joining and leaving these multicast groups, which however, can result in a large amount of signaling traffic in a dynamic network. Further, the adaptation is limited to available layers. To solve this limited layer problem, proxy streaming systems cache video content at local proxy disks and transcode (decode/recode) the video bitstream for different users, which may cause delay and huge computational burden if they serve a large number of users. Meanwhile, our method only sends one bitstream but can efficiently adapt video frame rate, quality, and spatial resolution for diverse users without transcoding. Moreover, FGA-FEC can provide an efficient scalable error-control mechanism in cooperation with the scalable video coding.

This chapter is organized as follows. In Section 3.2, we describe the details of method. Simulated and experimental results are given in Section 3.3. Conclusions are given in Section 3.4.

3.2 Fine Grain Adaptive FEC

3.2.1 System Overview

We use an overlay infrastructure to assist video streaming to multiple users by providing lightweight support at intermediate overlay nodes. These overlay nodes

with certain service functions, such as bitstream adaptation, network monitoring and so on, more than just store-and-forward, are called data service nodes (DSN). Diverse end users may have different network connection, computational capacity, and video display size, hence, they probably have different subjective ideal video and adaptation order preferences. Here, *ideal video* is defined as the type of bitstream the user initially requests from the system. Additionally, *adaptation order* is the user's chosen adaptation order in terms of quality, frame rate and resolution. The ideal video and adaptation order are input parameters to the system from the user-node console at the beginning of streaming. Since DSNs are often placed within a high-speed network, in this study, we assume that there is *no congestion between DSNs*. We also assume that a profile (description file [108]) of the video bitstream is sent to DSNs before the streaming session starts. The computation and adaptation unit in this study is one GOP.

We outline our idea with a simplified example. In Fig. 3.1, DSNs construct an overlay network to serve several users. Users "A" to "G" have different ideal video preferences (shown as "frame-rate/resolution/bitrate"). Here C and Q represent the common CIF and QCIF formats, respectively, and p_a to p_g are the average packet-loss rates of the overlay virtual links.

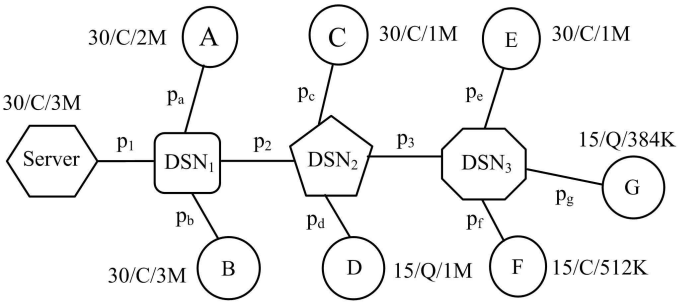


Figure 3.1: Intermediate adaptation of the video bitstream according to user video requests and network conditions by overlay data service nodes

Before streaming, each user signals its ideal video, adaptation order and minimum tolerable (acceptable) video in terms of frame rate, resolution and quality, and then sends the information to its parent DSN. While streaming, the DSNs determine a *video request* for each of their child users based on user's available bandwidth, ideal video, and adaptation order using Algorithm 1 below, on a GOP basis. Here, video

request is defined as the video bitstream that the DSN will request for a certain user from server or a parent DSN. These video requests and packet-loss rates of each link are collected and aggregated from end users to the server by the DSNs. The server then encodes the scalable video using FGA-FEC based on the highest video request (other less aggressive methods are possible, eg. highest quality of 90% of users, 80% of users etc) and the current aggregated packet-loss rates using Algorithm 1. FGA-FEC divides each GOP of the bitstream into small blocks and packs the FEC coded bitstream in such a way that if any original data blocks are adapted (actively dropped), the corresponding parity bits are also completely removed. At intermediate DSNs, adaptation is conducted by removing some blocks from each packet and/or dropping whole packets in concert with end users requests and network conditions (described in more detail at Section 3.2.4). Since there is no video and FEC decoding/recoding, FGA-FEC is very efficient in terms of computation. Furthermore, the data manipulation is at block level, which can be precise in terms of adaptation, given a sufficiently small block size.

Three important questions need to be answered.

- How should the video be encoded by FGA-FEC to facilitate multidimensional adaptation?
- How should FGA-FEC be designed to accommodate heterogeneous users, and yet be easily adaptable with highly scalable bitstreams at intermediate DSNs?
- How should the FGA-FEC encoded bitstream be adapted efficiently, given the user's ideal video, adaptation order, and network conditions?

3.2.2 Video Coding and Adaptation

In this section, we focus on how to encode and represent a scalable video to facilitate simple and precise adaptation of the bitstream to the available bit budget, both at the source and inside the network. This can be considered as the simplest case of FGA-FEC, where there is no loss inside network, hence no FEC is added to source data here. DSNs only need to adapt the source bitstream to satisfy diverse users.

In general, a piece of scalable bitstream which contains N nested tiers of scalability with the ith tier containing L_i layers, $i = 0, 1, 2, ..., N-1$, can be represented as an N-dimensional hypercube. The total number of *atoms* (elements) of the cube is $\prod_{i=0}^{N-1} L_i$. A specific atom is denoted $A(l_0, l_1, ..., l_{N-1})$, where $l_i \in \{0, 1, ..., L_i - 1\}$. For instance, Fig. 3.2 shows a cube of atoms in three dimensions {frame rate, resolution, quality} [44]. There are $L_0 = 4$ frame-rate layers, $L_1 = 3$ resolution layers, and $L_2 = 5$ quality layers. Each atom corresponds to a piece of subband bitstream with size in bits:

$$S(l_0, l_1, ..., l_{N-1}) = \|A(l_0, l_1, ..., l_{N-1})\|, \tag{3.1}$$

where $\|\cdot\|$ denotes the number of bits. Adaptation of the scalable bitstream is equivalent to selecting a subset of atoms for transmission.

Since we only use three dimensions of scalability: temporal (frame-rate), spatial (resolution) and SNR (quality), for simplicity, in the sequel, we use $\{L_t, L_s, L_q\}$ instead of the more general $\{L_0, L_1, L_2\}$.

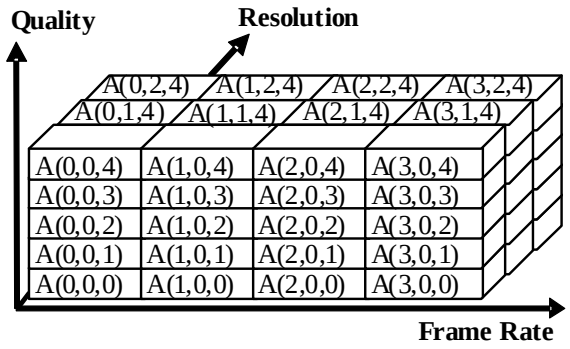


Figure 3.2: 3-D video scalability in the form of atoms of a GOP, $A(i, j, k)$ represents an atom of {frame rate, resolution, quality}

Given the representation in Fig. 3.2, we can adapt the atoms to the desired frame-rate, resolution and quality, according to a user's ideal video and adaptation order, as well as the available bit budget. Given a bit budget Ω[1], a subset of atoms

[1]In practice Ω is determined by the source bitrate permitted for the link, as determined in Section 3.2.3.

can be chosen to satisfy:

$$\sum_{l_t=0}^{T}\sum_{l_s=0}^{S}\sum_{l_q=0}^{Q(l_t,l_s)} S(l_t, l_s, l_q) \leq \Omega, \tag{3.2}$$

where T and S are the number of temporal and spatial layers, respectively, $T \leq L_t$ and $S \leq L_s$. Here $Q(l_t, l_s)$ is the number of quality layers at that temporal and spatial layer. There may be several different $T, S, Q(l_t, l_s)$ values which satisfy (3.2), so the particular set is chosen based on the user's ideal video and adaptation order. Since we have only three dimensions of adaptation, the total number of adaptation orders is $3 \times 2 \times 1 = 6$. Table 3.1 lists all the possible adaptation orders, where $\{t,$ $s,$ $q\}$ represents {frame rate, resolution, quality}. We will measure quality in terms of PSNR.

AO	Adaptation order	AO	Adaptation order
1	$q \rightarrow t \rightarrow s$	4	$t \rightarrow s \rightarrow q$
2	$q \rightarrow s \rightarrow t$	5	$s \rightarrow q \rightarrow t$
3	$t \rightarrow q \rightarrow s$	6	$s \rightarrow t \rightarrow q$

Table 3.1: Possible adaptation orders

The user can choose to adapt downwards SNR (Quality), frame rate, and resolution in any particular order. For example, the user's ideal video may have PSNR no less than γ dB with full frame rate and spatial resolution, and his/her chosen adaptation order is $q \rightarrow t \rightarrow s$. If the desired PSNR cannot be met, we reduce the frame rate down one temporal level and, if necessary, further reduce resolution down one spatial level, and do this iteratively, until to the minimum tolerable bitstream with $L_{t\,\min}$ temporal levels and $L_{s\,\min}$ spatial levels, or the minimum PSNR (quality) requirement is met. If the user's minimum quality can not be met, then no video is sent to that user.

Table 3.2 summarizes some terms that are used in the sequel.

To respond to the available bit budget Ω, DSNs adapt the bitstream based on users' specified adaptation orders. At each spatial/temporal adaptation step, we determine the best achievable video quality (maxPSNR), and iterate until satisfying

Terms	Definitions
L_t,L_s,L_q	the number of layers in user's ideal video in terms of temporal, spatial and quality.
AO	adaptation order identification number
γ, $L_{t\,\min}$, $L_{s\,\min}$	adaption order parameters: PSNR $\geq \gamma$ dB, temporal layers $\geq L_{t\,\min}$, and spatial layers $\geq L_{s\,\min}$
T, S, Q	the number of layers in user's video request in terms of temporal, spatial and quality
maxPSNR	best achievable video quality at a certain spatial and temporal layer, given the available bit budget
B	available bandwidth of a link
p	packet-loss rate of a link
$E[D(R)]$	mean video distortion
Ω	bit budget allocated to source data

Table 3.2: Terms used in the algorithm descriptions

maxPSNR $\geq \gamma$ dB according to the specified adaptation order:

$$\begin{aligned} \text{At each step}: \quad & T = i; \quad i \in \{L_t, L_t - 1, \cdots, L_{t\,\min}\} \\ & S = j; \quad j \in \{L_s, L_s - 1, \cdots, L_{s\,\min}\} \\ \text{Find}: \quad & \text{maxPSNR}(T, S) \\ \text{Subject to}: \quad & \sum_{l_t=0}^{T} \sum_{l_s=0}^{S} \sum_{l_q=0}^{Q(l_t,l_s)} S(l_t, l_s, l_q) \leq \Omega \end{aligned} \tag{3.3}$$

A unique solution of $T, S, Q(l_t, l_s)$ can be found by a simple search along the specified adaptation order. The resulting data set could be the same as the user's ideal video, given enough available bit budget, or it could shrink down to zero if a user's minimal requirements cannot be met due to very low available link bit budget.

We use the fully scalable MC-EZBC video coder [28] to show the method of adaptation. One bitplane of the embedded MC-EZBC coded bitstream is shown in Fig. 3.3, employing 3-level MCTF over an eight-frame GOP as an example. After the temporal decompositions, we obtain 8 temporal subband frames: 1 t-LLL frame, 1 t-LLH frame, 2 t-LH frames and 4 t-H frames. For these 8 frames, we do a 3-stage spatial decomposition for each frame and encode the generated spatial subbands

with EZBC, bitplane by bitplane.

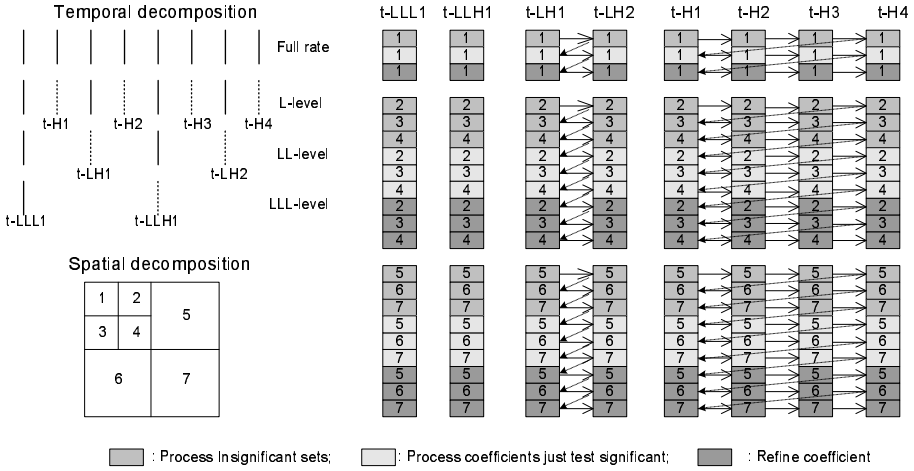


Figure 3.3: One bitplane of an 8-frame GOP. The subband coding passes inside a GOP are interleaved to achieve approximate equal significance across time.

Since we use an approximately orthonormal transformation both in spatial and temporal domain, in the absence of many unconnected pixels, the corresponding bitplanes of different subbands should have about the same importance. For each spatial subband, there are three passes: process insignificant sets, process coefficients just tested significant and refine the coefficients [28]. The subband coding passes are independently addressable in the bitstream and can be interleaved to facilitate adaptation. The resulting encoded GOP can then be illustrated as shown in Fig. 3.2, where for example, each atom could correspond to one encoded subband bitplane. Adaptations of the encoded bitstream can be achieved by choosing an appropriate set of subbands. For example, if a user wants to view a video with half frame rate, we choose all subbands of temporal frames t-LLL1, t-LLH1, t-LH1 and t-LH2. Similarly for half frame rate and half resolution, we can further remove subbands corresponding to spatial levels 5, 6, 7 from the half frame rate bitstream. Quality (PSNR) scalability can be achieved by recursively coding bitplanes from the most significant bit (MSB) to least significant bit (LSB), until the source bit budget Ω is satisfied.

Fig. 3.4 illustrates a general bitstream hierarchy of an encoded GOP in 1-D format, where the encoding unit consists of independently decodable bitstreams

$\{Q^{MV}, Q^{YUV}\}$. Here,we use the same notation as [51]. Let $l_t \in \{1, 2, ..., L_t\}$ denote

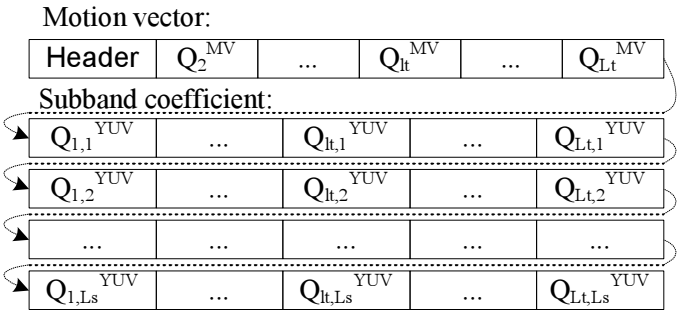


Figure 3.4: Hierarchy of MC-EZBC bitstream to facilitate 3-D adaptation

the temporal scale. The motion vector (MV) bitstream, Q^{MV}, can be divided into temporal scales and consists of Q_{lt}^{MV} for $2 \leq l_t \leq L_t$. Let $l_s \in \{1, 2, ..., L_s\}$ denote the spatial scale. The subband coefficient bitstream, Q^{YUV}, is also divided into temporal scales and further divided into spatial scales as $\{Q_{lt,ls}^{YUV}\}$, for $1 \leq l_t \leq L_t$ and $1 \leq l_s \leq L_s$. Thus, the video at $(1/4)^m$ spatial resolution and $(1/2)^n$ frame rate is obtained from the full bitstream as:

$$\begin{aligned} Q_{m,n} &= \{Q_{lt,ls}^{YUV} : 1 \leq l_s \leq L_s - m; 1 \leq l_t \leq L_t - n\} \\ &\bigcup \{Q_{lt}^{MV} : 2 \leq l_t \leq L_t - n\} \end{aligned} \tag{3.4}$$

In every sub-bitstream $Q_{lt,ls}^{YUV}$, subbands from Y, U and V are coded in an embedded manner from the MSB to the LSB. Scaling in terms of quality is obtained by stopping the decoding process at any point in bitstream $Q_{m,n}$, given the available bit budget.

Since the adaptation can be implemented as simple dropping of corresponding atoms along a pre-defined adaptation order, the DSNs do not need to decode and recode the bitstream, and are thus computationally very efficient.

3.2.3 Fine Granular Adaptation in the Presence of FEC

In this section, we move on a more general case where error control techniques are needed to protect the embedded bitstream from network losses. We focus on how to channel encode a scalable bitstream to facilitate adaption of both video bitstream

and error control codewords for diverse users without transcoding. This effectively extends the scalability to channel coding.

Automatic retransmission request (ARQ) and FEC coding are two widely used methods to protect packets from channel losses. Due to the feedback implosion problem in a multicast environment, we choose to study FEC as our protection method. When parts of the video bitstream are actively dropped, the DSNs need to update the FEC codes. This update has the same basic requirements as the source video adaptation - efficiency (low computational cost) and precision (if a part of the video data is actively dropped, parity bits protecting that piece of data should also be removed). Based on these considerations, we propose a precise and efficient fine grain adaptive FEC scheme based on Reed-Solomon (RS) codes and PET [36]. Arbitrary adaptation of RS codewords is difficult. For example, RS(n, k) codeword cannot be adapted to RS$(n - l, k - l)$ by simply dropping l symbols. One way to adapt an RS(n, k) is to decode first and then recode RS$(n - l, k - l)$, which is not computationally efficient for multiple adaptations along the transmission path or for multiple heterogeneous users. FGA-FEC solves the problem by adapting the FEC in a "fine granular" manner to satisfy multiple diverse users, as discussed below.

The FGA-FEC encoding method extends PET [36] and MD-FEC [39] by adding scalability (adaptation) features. Here we briefly overview the general idea of MD-FEC.

3.2.3.1 MD-FEC Overview

In a MD-FEC approach, one GOP of embedded bitstream is unequally protected by a sequence of erasure-correcting Reed-Solomon codes to produce N equally important descriptions, based on current network conditions. The bitstream is first divided into N sections, S_i, $(i \in [1, N])$ marked with rate breaking points $R_0, R_1, R_2, ..., R_N$ at Fig. 3.5, where $R_0 \leq R_1 \leq R_2 \leq ... \leq R_N$ and $R_0 = 0$, and the unit of rate is in bits per second (bps). Section i is further split into i equal size subsections (marked by $S_{i,1}, ..., S_{i,i}$) and encoded by a $RS(N, i)$ code.

RS codes are applied to each section vertically, the contributions from each of the N levels are then concatenated horizontally to form the N descriptions as

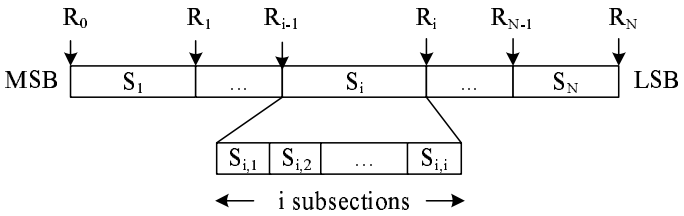


Figure 3.5: Rate partition of an embedded bitstream into N layers or quality levels, from most significant bit (MSB) to least significant bit (LSB)

shown in Fig. 3.6. In this study, one description is equivalent to one network packet, so we will use the terms "description" and "packet" interchangeably. This packetization scheme thus provides the property: if any i packets are received, decoding is guaranteed up to R_i. This is because sections protected by RS codes stronger than $RS(N, i)$ can all be recovered. Thus, we can say that all descriptions are equally important. After encoding, these generated descriptions can be sent in order from description 1 to description N.

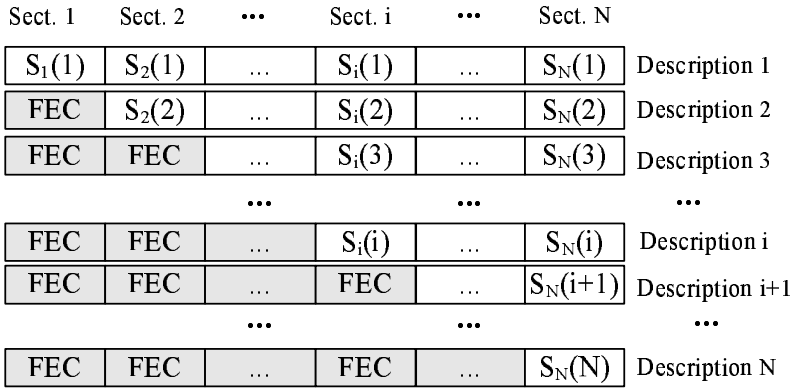


Figure 3.6: MD-FEC generates N descriptions or quality levels

3.2.3.2 FGA-FEC Encoding

FGA-FEC encoding considers not only the channel conditions, but also the user's video preference and predefined adaptation order, as well as the feasibility of intermediate adaptation. Given a GOP of source-coded video bitstream organized from MSB (R_0) to LSB (R_N), shown at the top in Fig. 3.7, suppose we want to encode this GOP into N descriptions, we first run an optimal bit allocation scheme according to network conditions and user's video preference/adaptation order, and

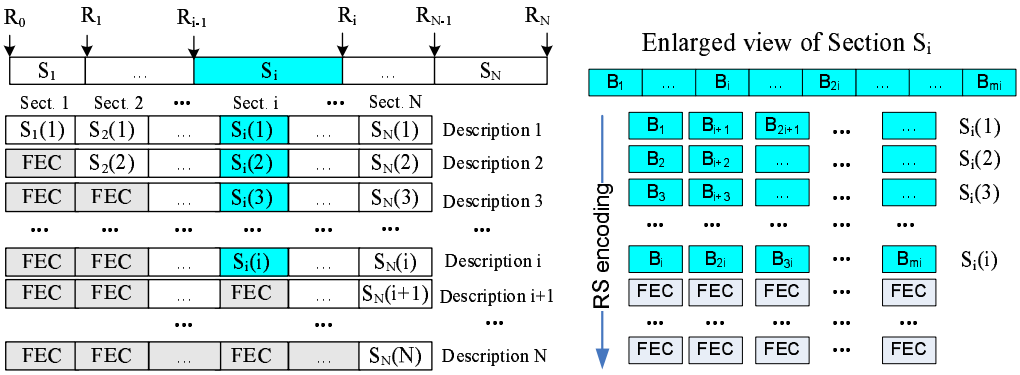


Figure 3.7: FGA-FEC encoding of one GOP. Here, FEC is added vertically at block level and each horizontal row of blocks is packetized into one network packet.

divide the bitstream into N sections S_i, $(i \in [1, N])$, marked with source-rate break points $R_0, R_1, R_2, ..., R_N$, where $R_0 \leq R_1 \leq R_2 \leq ... \leq R_N$ and $R_0 = 0$. Here R_N would correspond with Ω of (3.2) in Section 3.2.2. The total bitrate for all the descriptions must be less than B, the available bandwidth on the link. Section S_i $(i \in [1, N])$ is further split into equal size subsections with each subsection i blocks, therefore, the total number of subsections of S_i can be calculated as $m = \|S_i\|/(i \times \|block\|)$. These subsections are encoded by an RS(N, i) code vertically at block level to generate parity blocks.

Actually, it is not necessary that each section S_i, $(i \in [1, N])$ must have data in it. For example, in one of our experiments ($N = 64$, $p = 0.2$ and $B = 1$ Mbps in Fig. 3.15(a)), the source rate allocation result is $R_0 = R_1 = \cdots = R_{26} = 0$ and $R_{44} = R_{45} = \cdots = R_{64}$, i.e. only sections $S_i, i \in [27, 44]$ have data and the corresponding RS codes are RS$(64, i)$,$i \in [27, 44]$, respectively. After FEC encoding, each vertical column represents a data subsection divided into blocks, followed by the generated parity blocks. Each horizontal row of blocks is packed into one description, and in this study, one description is equivalent to one network packet. In Fig. 3.7, $S_i(j)$ denotes the bitstream of the ith section packed in jth description. Since each column is independently coded, we can adapt the bitstream by easily removing related columns. Obviously, a smaller block size means finer granularity and hence better adaptation precision. The criteria to choose the size of a block is based on the adaptation precision demand of users. In this study, we usually choose one byte

as the size of a block. This is possible the best block size due to adaptation of a network packet is much more efficient at byte level than at bit level, further, 8 bits is also a good symbol size for RS codes in networking. For example, if the size of each encode description is 1500 bytes and the size of a block is 1 byte, we can have 1500 layers for adaptation. The main encoding difference between MD-FEC and FGA-FEC is at the splitting and encoding of each section S_i, ($i \in [1, N]$). For MD-FEC, S_i is divided into i subsections (with size $\|S_i(i)\|$) and the RS(N,i) is added over these subsections vertically. Therefore, no bitstream adaptation is considered in MD-FEC, it is an end-to-end protection method. For clarity, we compare the encoding methods of MD-FEC vs. FGA-FEC at an example shown in Fig. 3.8.

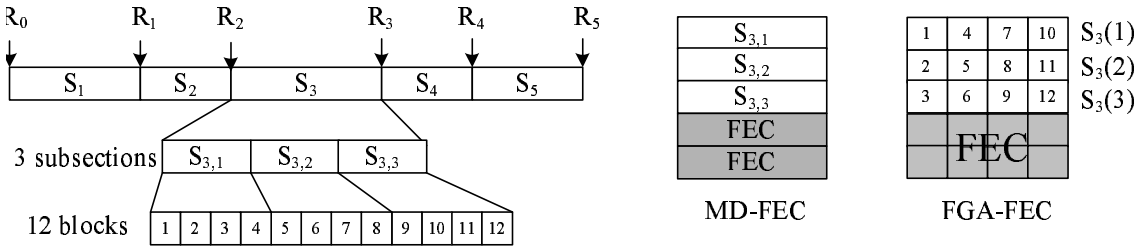


Figure 3.8: An example compares the encoding methods of MD-FEC vs. FGA-FEC, here we only show the encoding details of section 3, other sections should be similar

In this example, the number of descriptions is $N = 5$, the RS codes applied to section S_1, S_2, S_3, S_4, S_5 are $RS(5,1), RS(5,2), RS(5,3), RS(5,4), RS(5,5)$, respectively. Here we only show the encoding details of section 3 at Fig. 3.8, other sections should be similar. In MD-FEC, section S_3 is split into three equal size subsections, $S_{3,1}, S_{3,2}, S_{3,3}$. MD-FEC encodes this section by adding RS(5,3) code over the three subsections. On the other hand, FGA-FEC divides section S_3 into blocks (for simple, in this example, we arbitrarily choose 12 blocks for section 3, resulting 4 blocks for each subsection), and FEC RS(5,3) is applied vertically across these data blocks to generate parity blocks. Each vertical column represents a data section divided into blocks, followed by the generated parity blocks, as shown in Fig. 3.8, where $S_i(j)$ denotes the bitstream of ith section packed in jth description.

Similar to MD-FEC [39], FGA-FEC transforms the priority ordered bitstream from an embedded video coder into equal priority descriptions. If any i out of N

packets are received, the decoder can decode the bitstream up to R_i. In addition, FGA-FEC has added scalability (adaptation) features. The granularity of FGA-FEC adaptation is at block level. For instance, suppose that a DSN needs to adapt the video by dropping part of the bitstream, this can be achieved by removing some original data and FEC blocks related to that piece of bitstream from each network packet. Fig. 3.9 illustrates a possible adaptation of one FGA-FEC encoded GOP, where two blocks need to be removed from each description of the GOP. Hence, all

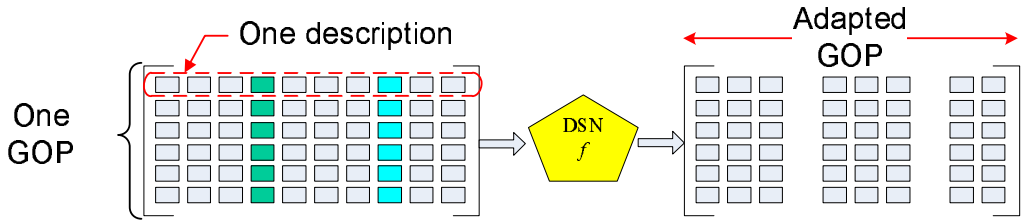


Figure 3.9: Adaptation of an FGA-FEC encoded GOP, two dark blocks are removed from each description, including both original data and parity bits.

packets (both data and parity) are shortened, and no FEC transcoding is necessary. Further, the removed parity bits correspond precisely to the video data bits that are dropped. Later on, we refer to this as the *direct truncation* method.

Next, we find the optimal FEC assignment for a given scalable video bitstream over a certain channel. The bitstream has three types of adaptation: SNR, frame-rate, and resolution. Frame-rate and resolution adaptation can only be performed in discrete layers. For instance, if the full video resolution in Fig. 3.3 is CIF, it can only be adapted to QCIF video in terms of resolution adaptation. The exclusive CIF-related parts of the bitstream (i.e bits related to subbands 5,6,7 in the example in Fig. 3.3) are directly removed from the original bitstream, and no optimization is needed. Similarly, dropping bits of subbands of all four L-level frames (t-H1, t-H2, t-H3 and t-H4) would result in half frame rate. Since the bitstream is progressively encoded, the SNR adaptation is fine granular. Then we need to find the optimal solution for SNR scalability. Furthermore, whenever frame-rate adaptation or resolution adaptation is performed, protecting the adapted bitstream remains the problem of finding an optimal protection in terms of quality.

Suppose we want to create N packets per GOP to serve a user with minimum

tolerable bitstream (γ, $L_{t\min}$, $L_{s\min}$) and the adaptation order (AO) over a channel with available bandwidth B. Following [39] and [55], let q_i be the probability that any i out of N packets are successfully delivered. The goal is to find the optimal bitrate partition $R = \{R_1, R_2, ..., R_N\}$ in Fig. 3.7 at a certain adaptation step, which minimizes the end-to-end mean distortion $E[D(R)]$ and the corresponding maxPSNR satisfies maxPSNR $\geq \gamma$ dB,

$$E[D(R)] = \sum_{i=0}^{N} q_i D(R_i), \tag{3.5}$$

subject to:

$$\begin{cases} 0 \leq R_1 \leq R_2 \leq ... \leq R_N; \\ R_{\text{total}} \leq B; \\ R_i - R_{i-1} = r_i \times i; \qquad r_i \geq 0, \ \forall \ i = 1, ..., N \end{cases}$$

where as mentioned earlier B is the available bandwidth for the link or channel, and $R_0 = 0$. Also r_i is the source rate of ith section S_i packed in each description. The last constraint item of (3.5) is to make sure that each section has an integral multiple of blocks. Given a packet-loss probability p and assuming independent losses, q_i can be calculated as:

$$q_i = \begin{pmatrix} N \\ i \end{pmatrix} (1-p)^i p^{N-i}. \tag{3.6}$$

Here R_{total} is the total bitrate (bandwidth) for both FEC and video data and is calculated as:

$$\begin{aligned} R_{\text{total}} &= \frac{R_1}{1}N + \frac{R_2 - R_1}{2}N + \cdots + \frac{R_N - R_{N-1}}{N}N \\ &= \sum_{i=1}^{N} \frac{N}{i(i+1)} R_i \ = \ \sum_{i=1}^{N} \alpha_i R_i, \end{aligned} \tag{3.7}$$

and $\alpha_i = \frac{N}{i(i+1)}$ for $i = 1, 2, \cdots, N-1$; with $\alpha_N = 1$. The total bitrate of a GOP is actually a sum up of the rate of each column in Fig. 3.7, including both FEC and original video data.

Following the adaptation order, at each adaptation step, finding the optimal source-rate break points $\{R_i, i \in [1, N]\}$ is effectively a bit allocation problem. Several algorithms are available in [39, 37, 38]. For simplicity, we use a generalized BFOS algorithm (Steps 1−5 in Algorithm 1 below) which finds the optimal bit allocation solution by a simple search in each adaptation step. As shown in [39], if $\alpha_i/q_i \leq \alpha_{i+1}/q_{i+1}$ for some i, then in the optimal solution we will have $R_i = R_{i+1}$. Hence, R_{i+1} need not be computed - it is sufficient to optimize R_i and then set $R_{i+1} = R_i$ at the end. We therefore remove from the list $R_1, R_2, \cdots, R_N$ any such R_{i+1}, remembering its indices, and re-label the remaining variables into a new list $R_1, R_2, \cdots, R_{N'}$, where $N' \leq N$. After the optimization, we insert these R_{i+1} indices back and have the result $R = \{R_1, R_2, ..., R_N\}$. We then calculate $E[D(R)]$ of this adaptation step and test if the corresponding maxPSNR satisfies maxPSNR $\geq \gamma$ dB. If so, solution found, otherwise move on to next adaptation level and repeat the process until we exhaust all adaptation levels. Algorithm 1 is run in both DSNs and server. The DSNs run Algorithm 1 to figure out user requests $\{T, S, Q\}$, while the server runs it to determine the FEC encoding $\{R_1, R_2, \cdots, R_N\}$. If no solution is found for a particular user at a DSN, no data is requested for this user. If no solution is found at server, no data is sent to network. The calculation of $\triangle D$ in Algorithm 1 is based on convex experimental R-D curve of the MC-EZBC encoded video bitstream. Given 1bps rate allocated from section i to section $i+1$, based on (3.5), the expected distortion increase contributed by section i is $q_i[D(R_i - 1) - D(R_i)]$, based on (3.7), the rate change is $\alpha_i R_i - \alpha_i(R_i - 1) = \alpha_i$. Similarly for section $i+1$, the expected distortion decrease is $q_{i+1}[D(R_{i+1}) - D(R_{i+1} + 1)]$ and the rate change is α_{i+1}. Therefore we can calculate the change in distortion/rate as following:

$$\begin{aligned}(\frac{\triangle D}{\triangle R})_i &= \frac{q_i[D(R_i - 1) - D(R_i)]}{\alpha_i} - \frac{q_{i+1}[D(R_{i+1}) - D(R_{i+1} + 1)]}{\alpha_{i+1}} \\ &= \frac{q_i}{\alpha_i}[D(R_i - 1) - D(R_i)] - \frac{q_{i+1}}{\alpha_{i+1}}[D(R_{i+1}) - D(R_{i+1} + 1)]\end{aligned}$$

More detail regarding the BFOS algorithm can be found in [4] and [5]. Given $p = 0$, Algorithm 1 is equivalent to solving (3.4) with no bit budget is allocated to FEC, i.e. the same as video adaptation in Section 3.2.2. Given the optimal

Algorithm 1: FGA-FEC optimization algorithm

Inputs: N, N', B, p, AO, γ, $L_{t\,\min}$, $L_{s\,\min}$;
Outputs: $\{R_1, R_2, ..., R_N\}$, T, S, Q;

Start:
1. For $i = 1, 2, ..., N'$, set $R_i = B$, calculate q_i, α_i;
Loop:
2. For $i = 1, 2, ..., N'$, calculate the change in distortion based on D-R curve at this adaptation level,

$$(\frac{\triangle D}{\triangle R})_i = -\frac{q_{i+1}}{\alpha_{i+1}}[D(R_{i+1}) - D(R_{i+1}+1)] - \frac{q_i}{\alpha_i}[D(R_i) - D(R_i - 1)] \quad (3.8)$$

Let l be the index i for which $(\frac{\triangle D}{\triangle R})_i$ is minimum;
3. Set $R_l = R_l - 1$.
4. Calculate the total rate R_{total};
5. If $R_{\text{total}} > B$, go to **Loop**, otherwise go to step 6;
6. For $i = 1, 2, ..., N$, round down R_i to the nearest multiple of i block;
7. Calculate $E[D(R)]$;
8. If $E[D(R)] \Leftrightarrow$ maxPSNR $\geq \gamma$ dB, solution found, **Stop**; otherwise move down one adaptation level per AO;
9. If adaptation level exhausted, **Stop**, no solution; otherwise, go to **Start**.

bit-allocation result, the rate break points of the bitstream are known, and we can encode the bitstream using FGA-FEC as illustrated in Fig. 3.7. Based on the encoded bitstream shown in Fig. 3.3, we also know the amount of FEC that should be applied to each atom. Therefore, we can re-group the atoms to facilitate a certain kind of adaptation. For example as shown in Fig. 3.10, with $L_s = 3$ and $L_t = 3$, we re-arrange the data for easier spatial and temporal adaptation, where $Q_{i,j}^{YUV}$ denotes sub-bitstreams corresponding to spatial level i ($i = 1$ corresponds to the lowest resolution) and temporal level j ($j = 1, 2, 3$ corresponds to temporal L-level, LL-level and LLL-level, respectively in Fig. 3.3), Q_j^{MV} $j = 1, 2$ denote the motion-vector bitstream at temporal level j . There are no motion vectors in the lowest temporal level (i.e. no motion vectors at LLL-level in Fig. 3.3). For temporal and spatial adaptation, DSNs can directly remove the appropriate parts from the

encoded bitstream in Fig. 3.10. For SNR adaptation, the DSN needs to calculate which sub-bitstreams need to be removed, and adapt each packet as illustrated in Fig. 3.9. Since we shorten each packet in the same GOP by removing related blocks, both FEC and data blocks are actively removed.

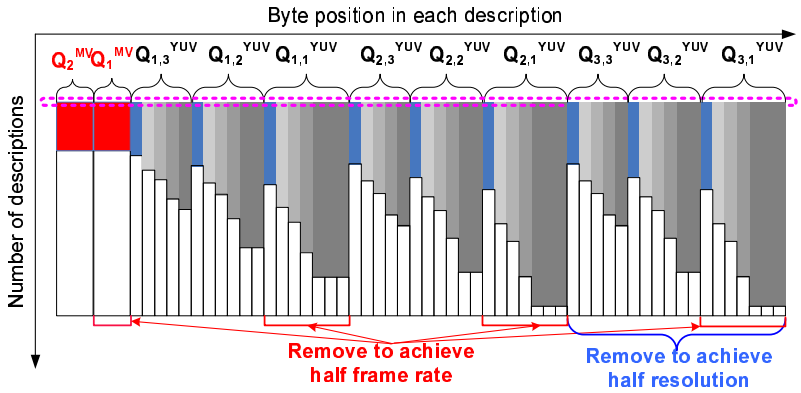


Figure 3.10: FGA-FEC encoded GOP can be re-organized to facilitate a certain kind of adaptation. Here, one horizontal line is one description and vertically it totals to N descriptions. Adaptation of PSNR can be easily achieved by removing related vertical blocks from each packet. White blocks contain FEC, colored blocks contains data.

To facilitate intermediate adaptation of a GOP, a header or information packet, describing how FGA-FEC encodes each GOP, is sent to the DSNs ahead of the coded and packetized video bitstream (Fig. 3.11).

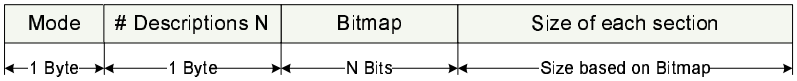


Figure 3.11: Information packet payload and the size of each field of the packet

In Fig. 3.11, *Mode* indicates the bitstream organization as temporal, spatial and SNR, *#Descriptions N* denotes the number of descriptions that will be encoded for one GOP, *Bitmap* denotes whether a bitstream section S_i, $i \in [1, N]$ has data (indicated as 1) or no data (indicated as 0), followed by the size of each non-zero section (occupying 2 bytes each). With this information packet and the bitstream description file, DSNs can then easily adapt the FGA-FEC encoded bitstream for diverse users. In this study, the block size of our scheme is set to 1 byte, a good symbol size for RS codes in networking. This information packet occupies very

little bandwidth. For example, to represent one FGA-FEC encoded GOP with 64 descriptions, its bandwidth consumption is less than 2 Kbps, given 2 GOPs per second (equivalent to 32 fps at 16 frames/GOP).

Table 3.3 summarizes the difference between MD-FEC and FGA-FEC.

	MD-FEC	FGA-FEC
FEC protection	Near optimal	Near optimal
Adaptation levels	SNR	SNR, Frame rate, Resolution
Adaptation by trans-coding	No	No
Adaptation methods	Dropping packets	Dropping packets and shortening packets
Adaptation precision	Section level, coarse	Block level, fine
Intermediate node computation	No, send as many packets as it can	Yes, to choose the best adaptation

Table 3.3: Summary of the differences between MD-FEC and FGA-FEC in terms of FEC protection and in-network adaptation

3.2.4 FGA-FEC Adaptation

In this section, we state how to adapt the FGA-FEC encoded bitstream according to user video request and network conditions. The adaptation unit is one GOP.

As mentioned above, each user node determines its ideal video, adaptation order and minimum tolerable bitstream, and then sends these parameters to their DSN before the streaming session as following:

$$\{L_t, L_s, L_q, AO, \gamma, L_{t\min}, L_{s\min}\},$$

where L_t, L_s, L_q denote the user ideal video in temporal, spatial, and quality layers, respectively; AO and γ, $L_{t\min}$, $L_{s\min}$ indicate the adaptation order and user minimal requirements, cf. Table 3.2.

While streaming, each end user periodically estimates the available link bandwidth B and packet-loss rate p of its uplink, where p can be measured based on observed packet losses, and B (in bps) can be estimated using the TCP friendly

equation [6],

$$B = \frac{S}{T_{rtt}\sqrt{\frac{2p}{3}} + T_{rto}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}, \tag{3.9}$$

where S is the packet size in bits, T_{rtt} is the estimated round-trip time between user and its DSN in seconds, T_{rto} is the TCP timeout on this link. Equation (3.9) was developed based on network experimental results to model TCP protocol and to estimate a TCP-friendly available bandwidth for a non-TCP network flow, such as UDP flow. If a non-TCP sender sends out packets at the estimated rate, the non-TCP flow would fairly share the network bandwidth with TCP flows.

Based on the estimated B and p, DSN runs Algorithm 1 to determine what it will request for this user, the result is T, S and Q, which denotes the user video request in temporal, spatial, and quality layers, respectively. Each DSN aggregates these video requests of its downstream users into a matrix, Θ, whose (i,j)th element $\Theta(i,j)$ denotes the requested highest quality layer at spatial resolution i and frame rate j among all its downstream users. The Θs are then aggregated along the DSNs back to the server. Suppose that a given DSN has two child DSNs that send their video requests Θ_1 and Θ_2, this parent DSN then produces aggregated video requests as follows:

$$\Theta(i,j) = \max(\Theta_1(i,j), \Theta_2(i,j)).$$

The packet-loss rate of each link is also aggregated and sent back to the server by the DSNs. Each DSN finds the largest packet-loss rate among its downlinks and then aggregates the result on its uplink, continuing back to the server. For example, in Fig. 3.1, the loss rate is aggregated as follows: DSN_3 finds the largest loss rate of its downlinks $\max(p_e, p_f, p_g)$, and then aggregates with its uplink loss rate p_3 as $p'_3 = 1-(1-p_3)(1-\max(p_e,p_f,p_g))$. At DSN_2, DSN_3 is one downlink of DSN_2 with packet-loss rate p'_3, the aggregated loss rate is $p'_2 = 1-(1-p_2)(1-\max(p_c,p_d,p'_3))$. At DSN_1, the aggregated loss rate to the server is $1-(1-p_1)(1-\max(p_a,p_b,p'_2))$. The server can then encode the video bitstream according to the aggregated user video request and loss rate. Effectively we are streaming through the overlay network to satisfy the maximum of the user video requests, but with a packet-loss rate that can produce an expected distortion at the users that is better than or equal to their

video requested value. Since we assume that there is no congestion between DSNs, the available bandwidth B on the backbone is assumed large enough so that R_N can accommodate the aggregated video requests. Our current work is devoted to removing this restriction.

Each DSN maintains a QoS parameter vector for both available bandwidth and packet-loss rate $\{\mathbf{B}, \mathbf{p}\}$ of its direct links, where

$$\mathbf{B} = \{B_i : i \in \{0, 1, 2, ..., N_{\text{down}}\}; \tag{3.10}$$

$$\mathbf{p} = \{p_i : i \in \{0, 1, 2, ..., N_{\text{down}}\}\}, \tag{3.11}$$

and $i = 0$ is for its uplink, N_{down} is the total number of its direct downlinks. These parameters would be used for adaptation of the FGA-FEC coded bitstream for each user.

While streaming, DSNs adapt the FGA-FEC coded bitstream for their direct downlinks based on user video requests, adaptation order, and network conditions. Adaptation to user video request in terms of resolution/frame rate is straightforward, DSNs can directly remove the bitstream and FEC codes as shown in Fig. 3.9 and Fig. 3.10. However, to adapt to a lower bandwidth B and loss rate p, DSNs need to do further adaptation of the bitstream by dropping descriptions (starting from description N and then $N-1, \cdots$, until $R_{\text{total}} \leq B$) or shortening each packet in Fig. 3.7 (starting from rate break points $R_N, R_{N-1}, \cdots$, until $R_{\text{total}} \leq B$) or by a combination of the two methods. We call this *FGA-FEC adaptation*. Suppose after video adaptation, there are only N' ($\leq N$) descriptions left with the updated rate break points $R_1, R_2, \cdots, R_{N'}$. Then the expected distortion after adaptation is:

$$E[D(R)] = \sum_{j=0}^{N'} q_j D(R_j), \tag{3.12}$$

subject to:

$$R_{\text{total}} \leq B,$$

where q_j is the probability of receiving j out of N' packets,

$$q_j = \begin{pmatrix} N' \\ j \end{pmatrix} (1-p)^j p^{N'-j}. \tag{3.13}$$

Here, $D(R)$ is the distortion-rate curve for the relevant spatial resolution and frame rate. We assume that these $D(R)$ curves at all spatial and temporal layers are available at server and represented by 21 points in terms of (Rate, Distortion). While streaming, the server sends these $D(R)$ curves to each DSN, these DSNs then convex interpolate the $D(R)$ curves to more points. For an instance, we usually interpolate a $D(R)$ curve to 2000 points in our optimization simulations.

Given the available bandwidth B and loss rate p of a particular user, there might be many pairs of N' and rate break points $\{R_i, i \in [1, N']\}$ that satisfy the bit budget. The DSN needs to find the best combination of dropping descriptions and shortening packets and uses the search Algorithm 2, FGA-FEC adaptation, to find a target adaptation bitstream that minimizes $E[D(R)]$. In detail, a DSN first adapts the bitstream to the user video request, since some subband bitstreams might be removed (i.e some columns removed as shown in Figs. 3.9 and 3.10), the rate break points $R_1, R_2, \cdots, R_N$ should be updated. If still $R_{\text{total}} > B$, the DSN needs to further adapt the bitstream to satisfy the available bandwidth, again considering the user adaptation order. At each spatial/temporal adaptation step, the DSN first finds the search range, by iteratively removing description N and rate break point R_N, description $N-1$ and R_{N-1}, and so on, until we are left with N' descriptions and rate break points that satisfy $R_{\text{total}} \leq B$. Since during transmission from server to the DSN, some packets, say δ out of N packets, might be lost or actively dropped by parent DSNs, the maximum number of descriptions which could be further dropped to satisfy the bit budget is $M = (N-\delta) - N'$. If $M \leq 0$, there is enough bandwidth to accommodate all packets, then all $N-\delta$ packets should be sent to the user, given video quality is satisfied, otherwise, move to next lower spatial/temporal adaptation level in user supplied AO. If $M > 0$, within the search range ($(N-\delta) \rightarrow N'$), we start from $I = N-\delta$ (all the received descriptions), and shorten the received descriptions by iteratively dropping the least important blocks

Algorithm 2: FGA-FEC Adaptation

Inputs : B, N, T, S, Q, AO, δ, γ, $L_{t\min}$, $L_{s\min}$
Outputs: $R_1, \cdots, R_{N'}$, I

Start:
Adapt bitstream based on user video request, T, S, Q;
Update $R_1, R_2, \cdots, R_N$;
If $R_{\text{total}} \leq B$ and max$PSNR \geq \gamma$ dB, forward all packets, **stop**,
otherwise, go on;
Loop:
Determine search range, iteratively drop descriptions to yield N' such that $R_{\text{total}} \leq B$;
for $(I = N - \delta;\ I \geq N';\ I--)$ **do**
  Remove descriptions N to I in Fig. 3.7;
  Shorten each description such that $R_{\text{total}} = \sum_{j=1}^{I} \alpha_j R_j \rightarrow B$
  Calculate q_j as (3.13) with $N' = I$;
  $E[D(R)]_I = \sum_{j=0}^{N'} q_j D(R_j)$;
end
min$E[D(R)]$ = min$\{E[D(R)]_I,\ I \in [N - \delta,\ N']\}$;
if (min$E[D(R)] \Leftrightarrow$ max$PSNR \geq \gamma\ dB$) **then**
  solution found, **stop**;
else
  Move down one adaptation level;
  If adaptation level exhausted, **stop**, no video is sent;
  Adapt bitstream based on this adaptation level;
  Update $R_1, R_2, \cdots, R_N$;
  If $R_{\text{total}} \leq B$ and max$PSNR \geq \gamma$ dB, forward all packets, **stop**,
  otherwise, go to **Loop**;
end

of Fig. 3.7 until we satisfy $R_{\text{total}} = B$, and then calculate $E[D(R)]_I$. Then we move to $I = N - \delta - 1$ (drop one received description), shorten each description to satisfy the bit budget, calculate $E[D(R)]_I$, and so on until $I = N'$. Thus finding the minimum distortion (min$E[D(R)]$) of this step. This process is repeated along the adaptation order, until we meet the user minimal quality requirement maxPSNR$\geq \gamma$ dB or exhaust all adaptation levels. After calculation, the DSN only needs to send out I updated descriptions, where I corresponds to the step with min$E[D(R)]$.

Algorithm 2 can be simplified to a coarse, computationally efficient method, we call direct truncation, wherein the DSN adapts the FGA-FEC coded bitstream by directly shortening each description to satisfy the available bandwidth, with no

descriptions further dropped ($I = N - \delta$). Direct truncation could be used at DSNs that lack computational power, such as battery powered mobile nodes.

Summarizing, the FGA-FEC scheme has the overall procedure shown in Table 3.4. Steps 1 and 2 are inputs to the streaming system from the users at start up, while steps 3-6 are repeated at every GOP.

Overall FGA-FEC procedure	
1.	Users decide adaptation-order, their individual input is AO, γ, $L_{t\,\min}$, $L_{s\,\min}$.
2.	Users decide ideal video, their input is L_t, L_s, L_q.
3.	DSNs determine video requests using Algorithm 1, the result is T, S, Q for each user.
4.	DSNs collect/aggregate video requests and loss rate to server.
5.	Server runs Algorithm 1 to allocate FEC and then encodes video in FGA-FEC format.
6.	DSNs adapt FGA-FEC coded bitstream to serve users using Algorithm 2.

Table 3.4: Overall FGA-FEC procedure

3.3 Simulations and Experiments

We distribute video coding functions across both server and network and adapt both the video bitstream and error-control codes to satisfy multiple diverse users by simply adjusting the packet size and/or dropping related packets at intermediate nodes, without decoding/recoding the data or FEC. Several questions need to be answered about the new technique:

1. Can the in-network block-based adaptation of embedded bitstreams achieve almost the same quality as source coding?

2. Since FGA-FEC is a generalization of MD-FEC, how does its performance compare with MD-FEC?

3. FGA-FEC adaptation serves multiple heterogeneous users by adapting both source- and channel-coded bitstream. Is it optimal or near optimal?

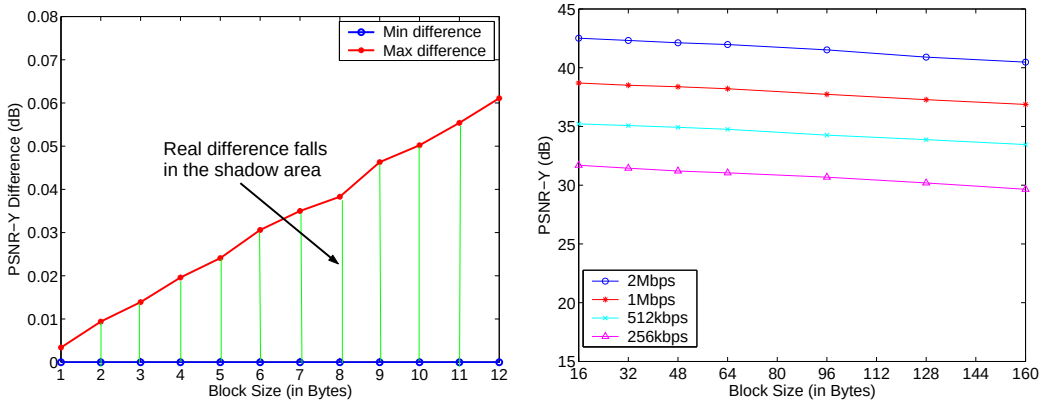
4. How does FGA-FEC perform compared with conventional Unicast streaming?

5. How does FGA-FEC perform in a multicast scenario compared with MD-FEC transcoding and layered multicast?

We performed simulations and experiments using the test sequence: *Foreman* CIF, 300 frames at 30 fps, and 16 frames/GOP. The scalable source coder MC-EZBC, and Reed-Solomon codes are employed. Adaptations are done at intermediate overlay nodes using the FGA-FEC encoded MC-EZBC bitstream. Each simulation is run at least ten times, and we present only averages for statistically meaningful results.

3.3.1 FGA-FEC vs. Source Coding

If there is no loss inside network, FGA-FEC encoding and adaptation is only performed over source data. At an intermediate DSN node, suppose we need to adapt the bitstream to match the available bandwidth of a certain downlink. The scalable source coder can generate a bitstream that exactly matches the bandwidth. But FGA-FEC adapts the bitstream at block level, so the adaptation will not be as precise as that at the source coder. So here we focus on comparing the coding efficiency of FGA-FEC vs. source coding, and we assume there is no FEC added.



(a) (b)

Figure 3.12: (a)Effect of block size, smaller block size means finer granularity of adaptation; (b)Effect of larger block size at different rate.

Given a video bitstream at 1 Mbps, FGA-FEC can packetize the bitstream into 128 packets per GOP in the order of SNR scalability, with block size of 8 bytes,

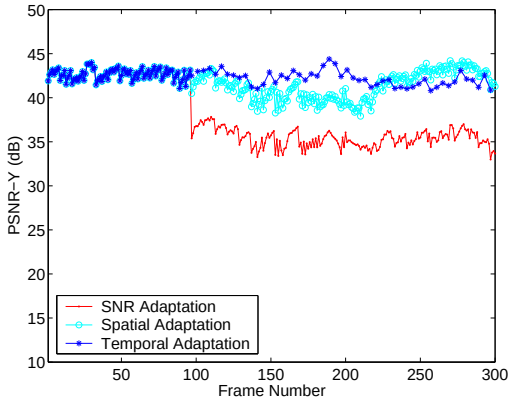


Figure 3.13: Video quality of 3-D adaptation to match the available bandwidth from 2 Mbps down to 512 Kbps

resulting in a network packet size of about 520 bytes. In a scenario where the last mile available bandwidth of a user is 991 Kbps, to match this bandwidth, FGA-FEC can only adapt the 1 Mbps bitstream to 976 Kbps by removing the last 3 blocks from each packet. Therefore, the overall PSNR of FGA-FEC is 0.04 dB lower than source coding in this case, due to the coarse FGA-FEC adaptation at block level than source coding at bit level.

Obviously, the block size of FGA-FEC can affect adaptation precision. In Fig. 3.12(a), we plot granularity of adaptation versus block size. Here, the last block is removed from each network packet (equivalent to remove the last block column in Fig. 3.7). Clearly, smaller block size means finer granularity. The two curves illustrate two extreme cases. The "min difference" happens in the case where both FGA-FEC and source coder remove the same amount of bitstream to satisfy a bit budget. The "max difference" happens at the case where FGA-FEC removes one block from each description in a GOP, while the source coder only needs to remove one block from a column in the bitstream. In other cases, the PSNR difference falls in the lined area between these two extreme curves. Fig. 3.12(b) further shows the adaptation granularity at larger block sizes and different bitrates by removing the last block of each packet. The granularity becomes coarser as block size becomes larger.

Each user has an adaptation order to respond to dynamic network conditions. In Fig. 3.13 we show the corresponding video quality when the available bandwidth

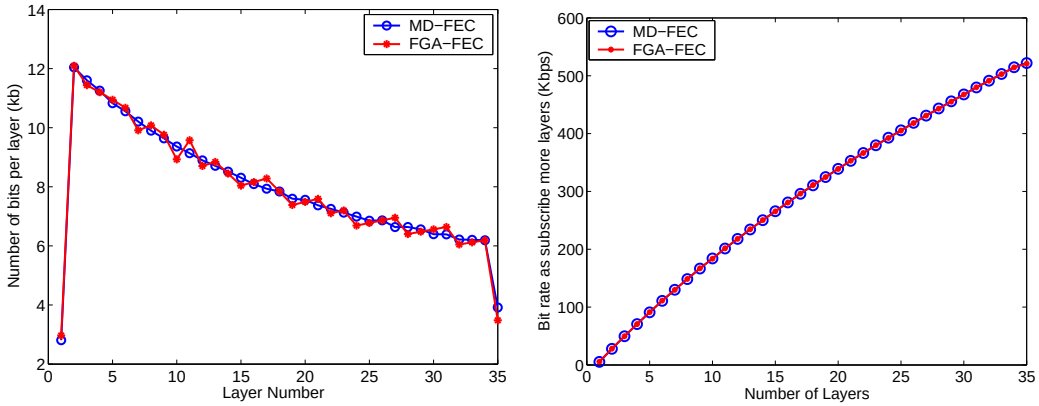
drops. Originally, the user is receiving a 2 Mbps, *Foreman*, CIF, 30 fps bitstream. Starting with frame 100, however, the user has only 512 Kbps available bandwidth. Here, we list three possible choices for the user: (a) SNR adaptation to 512 Kbps; (b) Temporal adaptation to 7.5 fps; (c) Spatial adaptation to QCIF. Both choices (b) and (c) need additional SNR adaptation to fit in 512 Kbps. The user can choose an adaptation order based on quality profiles like Fig. 3.13 and its own display and computing capabilities.

Results in this subsection show that FGA-FEC can adapt the bitstream almost as precisely as can the source coder. The maximum difference is less than 0.003 dB for a block size of 1 byte in Fig. 3.12(a).

3.3.2 FGA-FEC vs. MD-FEC Encoding

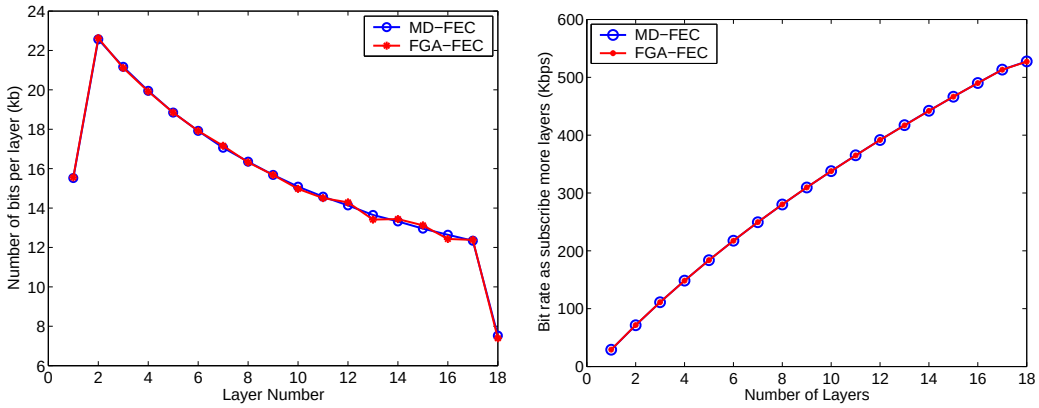
FGA-FEC extends MD-FEC by providing additional multidimensional adaptation capabilities with both source data and FEC data, to facilitate in-network processing. We compare the two methods by encoding the first GOP of *Foreman* into 128 and 64 descriptions, using both methods. The block size of FGA-FEC is set to 1 byte. Available bandwidth B is 1 Mbps and the average loss rate is 20% for both schemes. After the optimal bit allocation, the total number of encoded layers is 35 (128 packets case, where $R_0 = R_1 = \cdots = R_{52} = 0$ and $R_{87} = R_{88} = \cdots = R_{128}$) and 18 (64 packet case, where $R_0 = R_1 = \cdots = R_{26} = 0$ and $R_{44} = R_{45} = \cdots = R_{64}$) in the two cases of both schemes. Here, we refer to the piece of bitstream between two rate break points, shown at top of Fig. 3.7, as one layer.

Fig. 3.14(a) and 3.15(a) compare the numbers of bits in each layer for MD-FEC and FGA-FEC. We see that both algorithms can generate very similar layer sizes. The small visible difference is due to rounding at different precision levels (step 6 in Algorithm 1). For MD-FEC, R_i is rounded down to a multiple of i at the bit level, but for FGA-FEC, rounding is at block level, one byte. Also, smaller N corresponds to better (smoother) bitstream round-up precision. This is why the two methods perform even closer in Fig. 3.15(a) than in Fig. 3.14(a). We also show in Fig. 3.14(b) and 3.15(b) that FGA-FEC and MD-FEC generate almost the same bitrate as we move through the layers from base layer on upwards.



(a) Number of bits in each layer (b) Bitrate as layer adds up

Figure 3.14: FGA-FEC vs. MD-FEC in terms of bit allocation, GOP 1 of *Foreman* is packetized into 128 packets



(a) Number of bits in each layer (b) Bitrate as layer adds up

Figure 3.15: FGA-FEC vs. MD-FEC in terms of bit allocation, GOP 1 of *Foreman* is packetized into 64 packets.

3.3.3 FGA-FEC vs. MD-FEC - Multiple Heterogeneous Users

We next compare FGA-FEC vs. MD-FEC to satisfy diverse users with different bandwidth, by sending the encoded 64 descriptions to users with bandwidth ranging from 200 Kbps to 1000 Kbps as shown in Fig. 3.16, where DSN adapts the encoded GOP to response to different bandwidth. For simplicity, link packet-loss rate is set to 10% (uniformly distributed over 9-11%) for all users and there is no congestion between DSN and server, hence the loss rate in this backbone link is set to zero. We arbitrarily set PSNR=0 if there is not enough bandwidth for even the base layer. Fig. 3.17(a) shows the observed video quality (PSNR) of each user.

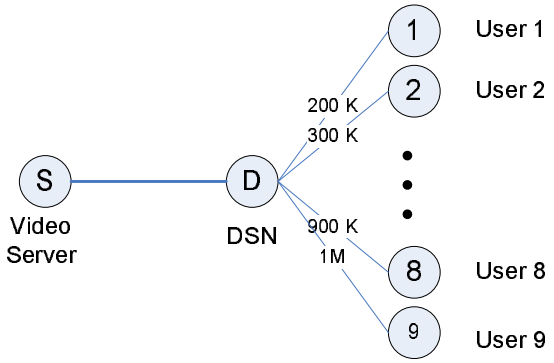
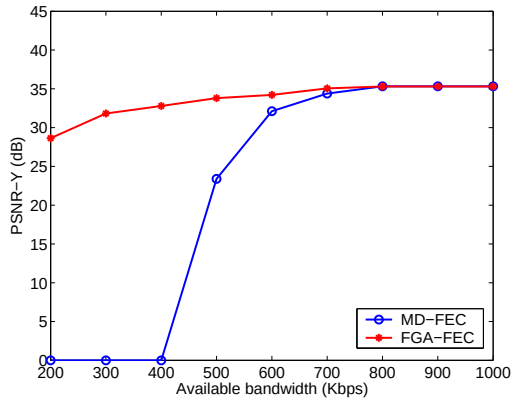


Figure 3.16: Network topology for comparison of FGA-FEC with MD-FEC. Diverse users connect through one DSN to a backbone link.

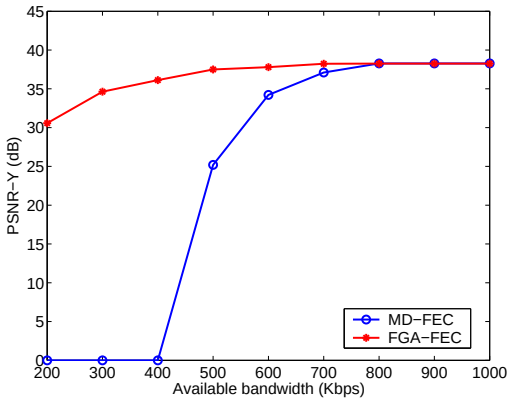
Clearly, FGA-FEC has much better performance in response to diverse user requirements. This is because MD-FEC adapts to dynamic bandwidth by sending as many packets as it can to the channel, hence some useless data is sent within packets. For example, if i out of N packets can go through a channel, the server can only decode up to rate break point R_i of the bitstream. Thus video data and FEC bits in the rate break point interval $[R_{i+1}, R_{i+2}, ..., R_N]$ cannot be successfully decoded but is still inside each packet, resulting in wasted bandwidth. On the other hand, FGA-FEC actively drops and shortens packets. Further, the adaptation is more precise than MD-FEC because all useless data is removed from each packet, thus saving some bandwidth for useful data and hence has better adaptation performance.

In addition to SNR adaptation shown at Fig. 3.17(a), FGA-FEC can do spatial and temporal adaptation as well. For example, at a certain time the users may want to view a QCIF in stead of CIF video to response to a smaller bandwidth, FGA-FEC can achieve this by adaptation using Algorithm 2 at the DSN node. Fig. 3.17(b) shows both spatial and SNR adaptation to different users.

Results in these two sections show that FGA-FEC performs almost the same as MD-FEC in terms of protection and bit allocation, but has a much better adaptation performance to network conditions. One reason is that MD-FEC was not designed for adaptation, it is simply an end-to-end protection method. It can coarsely scale in bitrate (quality) but not in frame rate and spatial resolution.

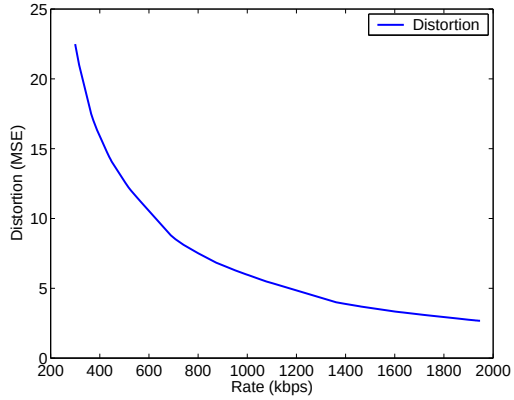


(a) SNR adaptation to different users

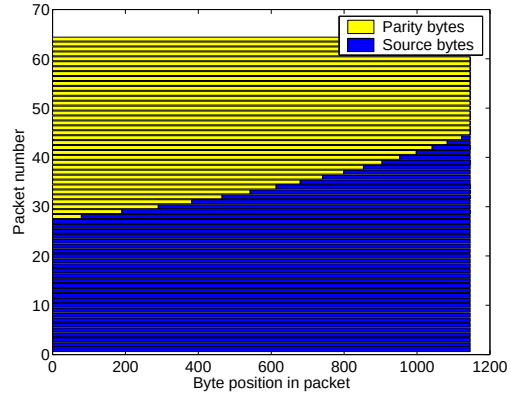


(b) Spatial adaptation to different users from CIF to QCIF

Figure 3.17: FGA-FEC vs. MD-FEC in terms of adaptation to different users with different bandwidth ranging from 200 Kbps (User1) to 1 Mbps (User 9).



(a) D-R curve of the seventh GOP of Foreman



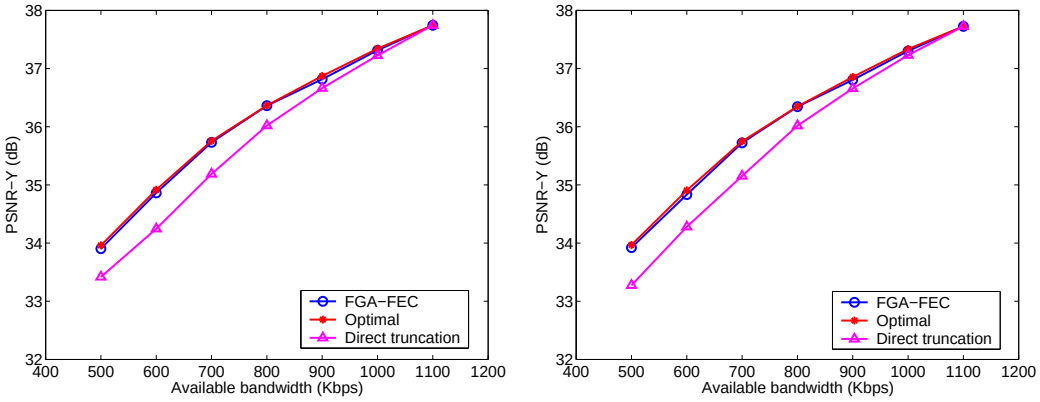
(b) Bit allocation 1100 Kbps, the source data and FEC data byte position in each 1145-byte packet

Figure 3.18: The operational rate distortion curve, and matched bit-allocation result of the seventh GOP *Foreman* CIF sequence

3.3.4 FGA-FEC Adaptation vs. Optimal Decode/Encode

At intermediate nodes, FGA-FEC adapts the bitstream and FEC by shortening packets and/or simply dropping packets via Algorithm 2. An upper bound solution would be to optimize the bitstreams at the DSN for each individual user and adapt the bitstream by decoding/recoding the FEC codewords based on user video request, user adaptation order, and network conditions. We call this method

optimal decode/recode solution.



(a) (b)

Figure 3.19: Comparison of FGA-FEC, optimal decode/recode solution, and direct truncation to serve users with different bandwidths; (a) against the theoretical mean distortion by calculation, (b) against the video quality by simulation

We compare FGA-FEC, optimal decode/recode solution, and the direct truncation method to satisfy users with available bandwidth ranging from 500 Kbps to 1100 Kbps. The network topology is the same as Fig. 3.16, but with different available bandwidth and the packet loss rate of each user is set to 15%. Encoded video bitstream is sent from server to each user through a DSN. FGA-FEC first optimizes the protection based on the highest bandwidth user (1100 Kbps) and the aggregated loss rate is 15%. The number of descriptions per GOP is set to 64. Given the rate-distortion curve of the seventh GOP of *Foreman* (Fig. 3.18(a)) and the distribution of the number of packets being received, the FGA-FEC encoded GOP (Fig. 3.18(b)) shows the byte position in each packet (refer to Fig. 3.7 for packetization). Fig. 3.19(a) compares the theoretical calculation results of mean PSNR of FGA-FEC adaptation using Algorithm 2, the optimal decode/recode solution, and direct truncation in response to various users with different network conditions. Algorithm 2 can both drop descriptions and shorten packets to achieve the best adaptation, which results in a near optimal video quality. The FGA-FEC adaptation is about 0.15 dB lower than the optimal solution in the mean distortion. The direct truncation method has a coarse, but still acceptable adaptation result.

Fig. 3.19(b) compares the resulting video quality of the three schemes by simulation, where the adapted bitstreams are transmitted to users through the topology to different users with different bandwidth.

FGA-FEC adaptation is only actively removing blocks within each packet instead of performing a complex FEC computation, so the computational burden is very low. We tested the FEC encoding/decoding time at a Pentium 4, 1.6 GHz machine running Linux 8.2. The task was to encode 115 packets with a size of 512 bytes each. To generate an RS(120,115) code, it took approximately 4 ms. We also tested the FGA-FEC adaptation burden on the same computer, the adaptation time to process the same number of packets was about 1×10^{-2} ms.

Results in this section show that FGA-FEC has near optimal performance in terms of error protection and adaptation, but much lower FEC computational burden than the optimal decode/recode solution that would be needed with just MD-FEC, since it has no data adaptation capability.

3.3.5 FGA-FEC vs. Unicast

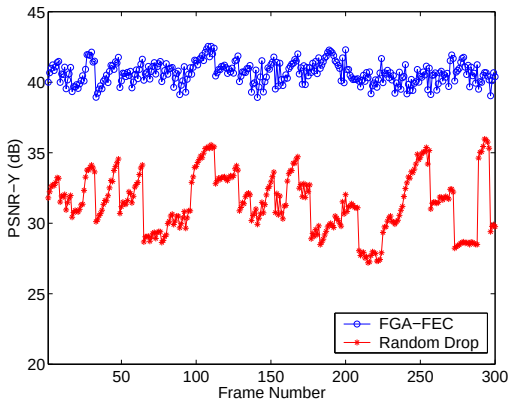


Figure 3.20: FGA-FEC vs. random drop to response to a congested link

Conventionally, when network congestion occurs, data packets are randomly dropped at the router to avoid congestion. On the other hand, FGA-FEC can adapt the packets at the intermediate nodes to reduce the bandwidth requirement, by dropping the least important parts of the bitstream. Given a 1.5 Mbps bitstream and available bandwidth of 1.455 Mbps, in Fig. 3.20 we compare PSNR-Y of FGA-FEC versus a random drop scheme with a 3% packet-drop ratio. There is no FEC

added in either scheme. Observe that the proposed scheme significantly outperforms random drop by about 10 dB. The reason for the large degradation of the random drop PSNR is the high dependency of the scalably coded video bitstream. If one packet is dropped, further packets in the same GOP become useless. Thus, the effective packet-loss rate is much higher than 3%.

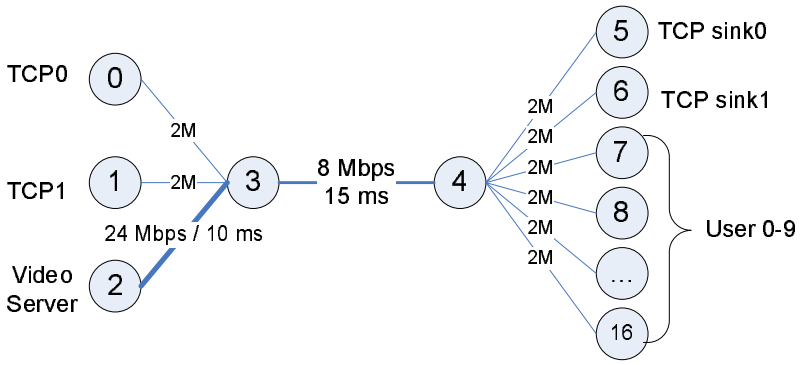


Figure 3.21: ns-2 topology for comparison of FGA-FEC with conventional unicast, the initial channel parameter set up is indicated at each link

3.3.6 Further Comparison via ns-2 Simulation

We further compare FGA-FEC versus unicast in terms of video quality using the network simulator ns-2 [96] for the architecture of Fig. 3.21, wherein twelve users are sharing a bottleneck between nodes 3 and 4. The ns-2 initial channel parameter set up is indicated at each link. Users 0 to 9 are requesting a scalable video from the server with the *ideal video source rates* and adaptation order *AO* shown in Table 3.5, the supporting network protocol is TFRC [6]. In FGA-FEC, we have designed the FEC using Algorithm 1 based on an available bandwidth of 2 Mbps and 2% packet-loss rate at the server. In Unicast, FEC is assigned based on actual available bandwidth and 2% packet loss rate. To increase the heterogeneity of the network, after streaming 96 frames (4 GOPs), the available bandwidths of User 6 and User 8 are dynamically changed to 800 Kbps and 600 Kbps, respectively, as shown in Table 3.5.

Due to congestion at the bottleneck in Unicast, each user fairly shares the bandwidth with others, including the two TCP users. Thus, the server can only stream video to each user according to its available bandwidth (not their ideal video

Users			**Supplied source rate (Kbps)**	
User	ideal video source rate (Kbps)	adaptation order AO	Unicast	FGA-FEC
0	1000	1	750	1000
1	1100	1	752	1100
2	1200	1	753	1200
3	1300	1	754	1300
4	1400	1	754	1400
5	1500	1	747	1500
6	1600	3	752	1600→800
7	1700	1	746	1700
8	1800	2	748→600	1800→600
9	1900	1	750	1900

Table 3.5: Network performance of using FGA-FEC vs Unicast

request), shown as *supplied source rate* in Table 3.5. Packets are actively dropped by the server according to their relative importance. In FGA-FEC, node 4 becomes a DSN node, it can adapt the bitstream to support different users. The required bandwidth between server and node 4 is 2 Mbps in this case. The total traffic at the bottleneck is at maximum 6 Mbps (2TCPs + 1TFRC from node 4), so there is no congestion in the FGA-FEC case. In Fig. 3.22, we show the captured video frames (93rd frame of *Foreman*) of the 9th user in Table 3.5. The effective throughput is 1.9 Mbps for FGA-FEC and 750 Kbps for Unicast. In this case, FGA-FEC is objectively 5.1 dB better than Unicast.

Fig. 3.23 compares PSNR of both FGA-FEC and Unicast of the 9th user at full frame rate and full resolution based on the actual available bitrate listed in Table 3.5. When the available bandwidth of User 6 is changed to 800 Kbps, the DSN needs to adapt the bitstream according to User 6's adaptation order and available bandwidth. Therefore, starting from frame 97, the DSN streams a half frame rate, full resolution video to User 6 with PSNR in Fig. 3.24(a). In response to User 8's network bandwidth change from 1800 Kbps to 600 Kbps, the DSN adapts the bitstream based on its available bandwidth and adaptation order AO, then a lower resolution (QCIF) bitstream is sent to User 8 starting from frame 97 as shown in Fig. 3.24(b). In both cases, Unicast still streams video in full frame rate and full

(a) FGA-FEC (b) Unicast

Figure 3.22: Sample video (93rd frame) of the 9th user in Table 3.5, given the available bandwidth, using FGA-FEC (a) and Unicast (b).

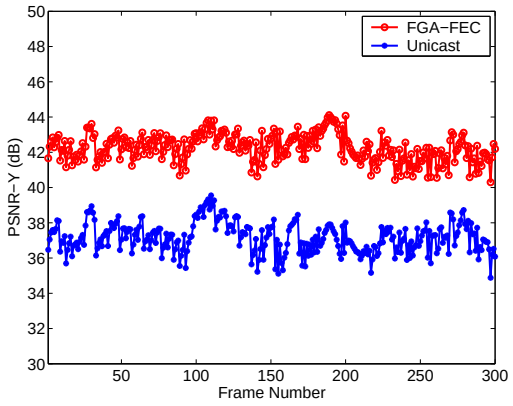
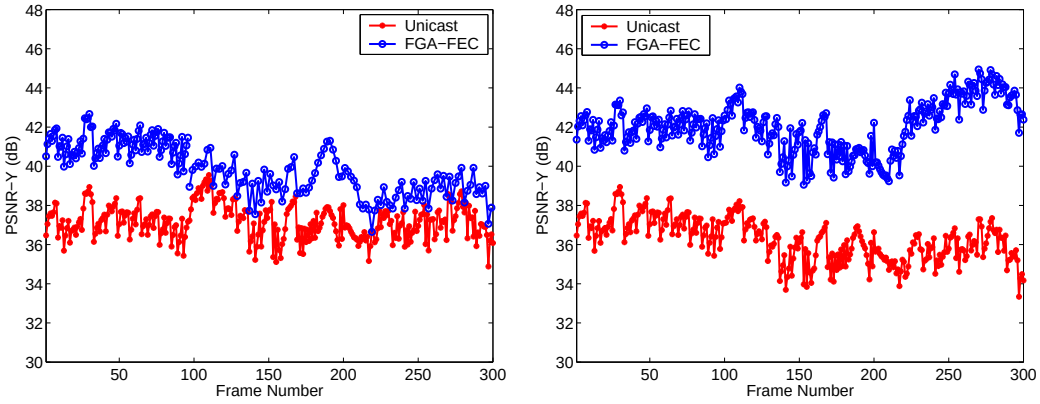


Figure 3.23: Comparison of PSNR of the 9th user: FGA-FEC vs. Unicast at full frame-rate and full resolution;

resolution.

3.3.7 FGA-FEC Adaptation vs. MD-FEC Transcoding in a Multicast Scenario

FGA-FEC adapts to network conditions by a combination of dropping packets and shortening descriptions. In a multicast scenario, this can be done at intermediate nodes to quickly response to different users. One optimal solution in terms of bitstream protection is FEC transcoding, i.e. intermediate nodes decode, re-optimize and re-encode FEC for diverse users. In this section, we explore how FGA-FEC adaptation performs in a multicast situation by comparing it with the



(a) FGA-FEC temporal adaptation vs. Unicast

(b) FGA-FEC spatial adaptation vs. Unicast

Figure 3.24: Comparison of FGA-FEC vs. Unicast in response to network bandwidth change starting at Frame 97: (a)FGA-FEC adapts the bitstream at half frame-rate and full resolution for User 6; (b) FGA-FEC adapts the bitstream at full frame-rate and half resolution for User 8.

optimal transcoding method. We choose to compare with a method called MD-FEC transcoding, proposed by Puri and Ramchandran [53] (Thanks for their generosity in providing their experimental code, so that we can have this section done fairly). We use the ns-2 network topology at [53] as shown in Fig 3.25. The protocol used here is a group formation protocol (GFP) [54]. The idea is to identify sets of receivers that share common bottlenecks in the multicast tree. These receivers have similar and correlated loss patterns and are hence grouped together. This approach ends up arranging the whole multicast tree as a hierarchy where groups downstream necessarily have worse reception than groups upstream of them and different groups are separated by bottleneck links. In the MD-FEC transcoding scheme, if the receiver group downstream conveys its channel state to the node acting as the transcoding agent, then that node can re-encode its received bitstream using MD-FEC encoder to optimally match the channel state below.

In both schemes, the server fully utilizes the available bandwidth, which is 1.64 Mbps. For MD-FEC, the transcoders for groups 2, 3, and 4 were placed at nodes 1, 2, and 10 respectively and the transmission profile (network conditions) for each group was passed to the corresponding transcoding agent. The transcoding nodes perform FEC decoding, re-optimization and recoding for each specific user according

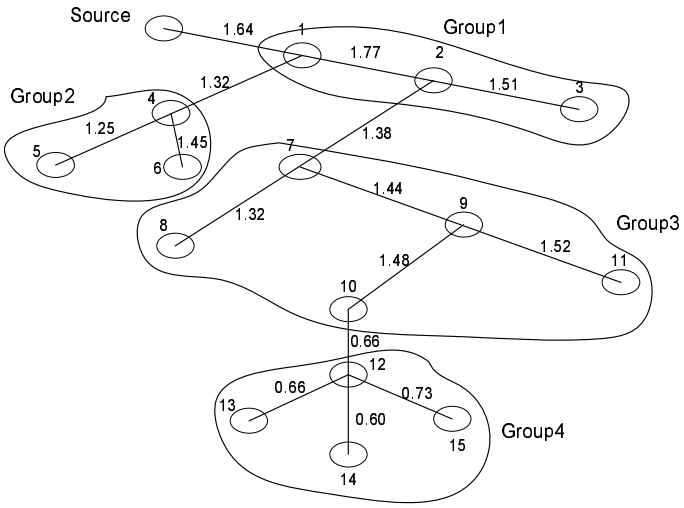
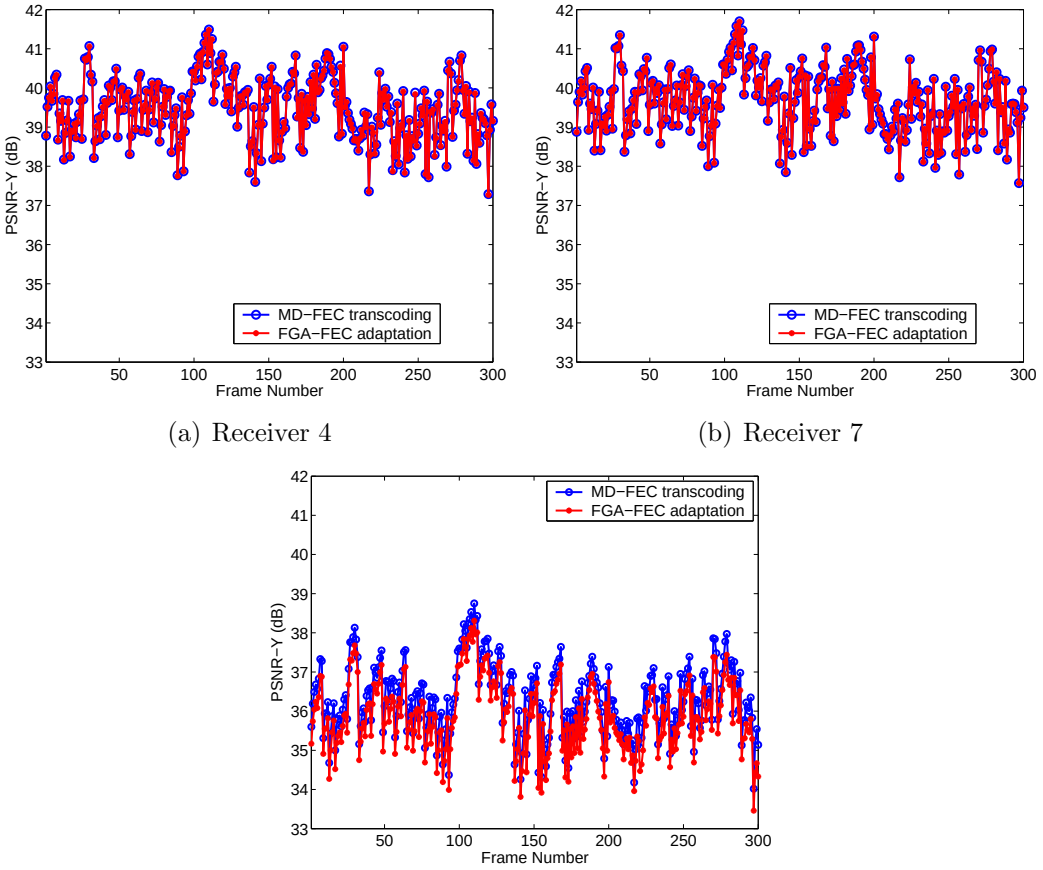


Figure 3.25: Network topology for a network of 16 nodes (link bandwidths are in Mb), the tree is organized into groups using GFP protocol.

to its transmission profile, thus, the total number of transcoding and optimization computation iterations is the same as number of different users. For our FGA-FEC, we optimize the FEC protection in the server and adapt the encoded FGA-FEC by dropping packets and shortening descriptions to match the available transmission profile at the same agent nodes (nodes 1, 2, 10 act as DSNs), without decoding and re-coding the FEC. In both cases, the sequence is *Foreman*, 16 frames/GOP, 30fp, the number of descriptions in one GOP is 64, and the block size in FGA-FEC is one byte. Therefore, the adaptation precision of FGA-FEC is one column in Fig. 3.7, which contributes to bitrate $64 \times 8 \times 30 \div 16 = 960$bps.

In Fig. 3.26, we compare the delivered quality in user 4 , user 7 and user 12. For user 4 and user 7, the available bandwidths are 1.32 and 1.38 Mbps, respectively. FGA-FEC can adapt the 1.64 Mbps encoded bitstream to 1.32 Mbps (1375 bytes/packet) and 1379.52 Kbps (1437 bytes/packet). Since these two adapted bitrate are close to the FGA-FEC encoding bitrate 1.64 Mbps, the adaptation is accurate and the quality is very close to MD-FEC transcoding, with FGA-FEC adaptation about 0.01 dB lower than MD-FEC transcoding in both cases. For user 12, the available bandwidth is 0.66 Mbps. FGA-FEC can adapt the encoded bitstream to 659.52 Kbps with 687 bytes/packet. Since the available bandwidth is far from 1.64 Mbps, the adaptation is not accurate, the quality is about 0.4 dB lower



(a) Receiver 4

(b) Receiver 7

(c) Receiver 12

Figure 3.26: Quality delivered (in dB) at various receivers. In receivers 4 and 7, FGA-FEC adaptation is about 0.01 dB lower than MD-FEC transcoding in both cases. In receiver 12, FGA-FEC is about 0.4 dB lower on average.

than MD-FEC transcoding.

FGA-FEC adaptation is achieved by actively dropping packets and shorting descriptions without any complex FEC transcoding. The computational burden is very small compared with MD-FEC transcoding. The detailed computational burden comparison is shown in Chapter 8 Section 8.3.4.

Results in this section show that FGA-FEC can perform almost as good as MD-FEC transcoding in terms of protection, but with much less computational burden in a multicast scenario.

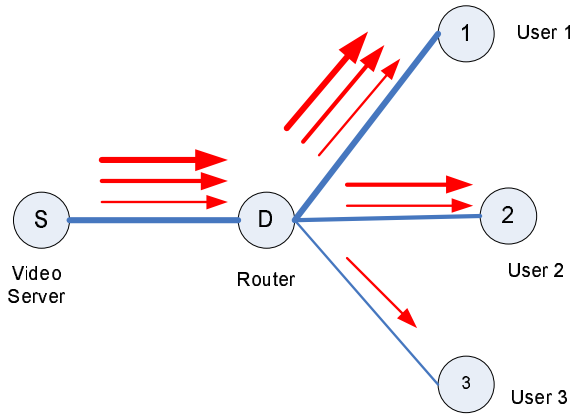
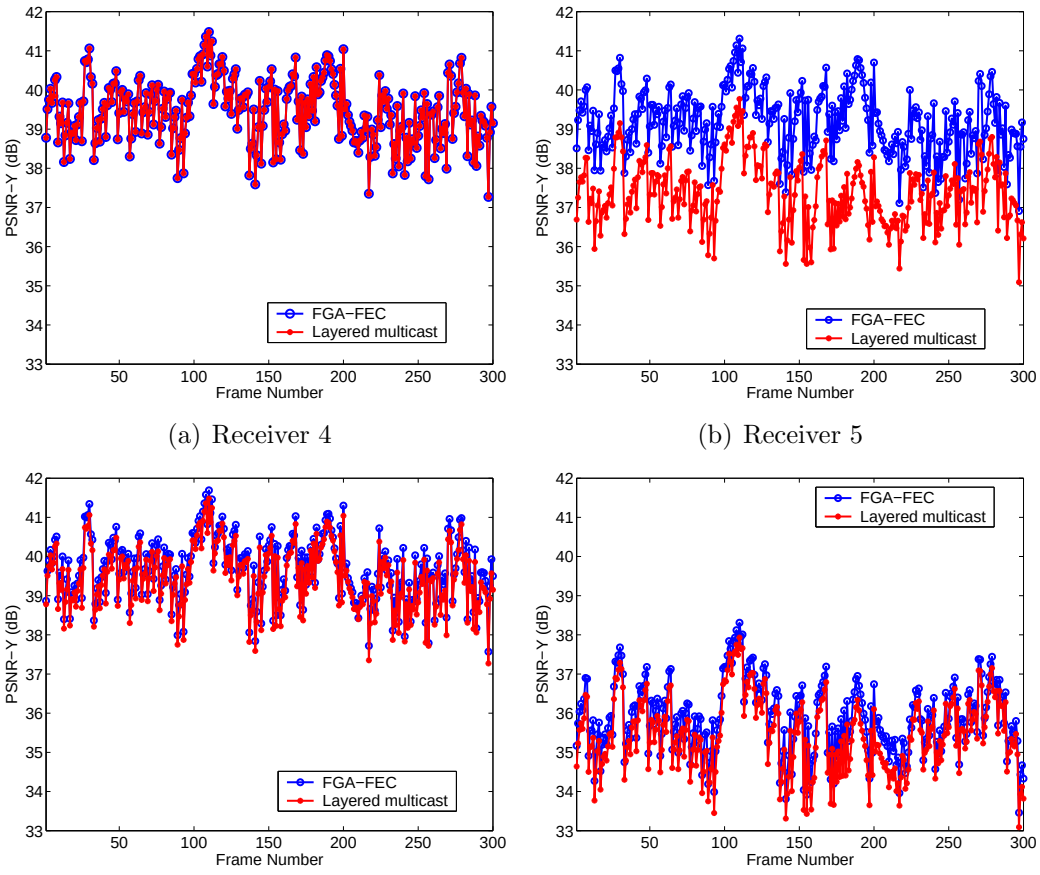


Figure 3.27: Layered multicast in an example of one server, three different users. The server sends out three video layers and users subscribe to different layers according to their network conditions.

3.3.8 FGA-FEC vs. Layered Video Multicast

In this section, we compare our FGA-FEC scheme with layered video multicast. A receiver driven layered multicast assumes a layered source that transmits different layers of multimedia stream to distinct multicast groups and each receiver subscribes to the multicast groups depending on its network characteristics [61, 69, 62]. An example is shown in Fig. 3.27, where the server sends out three video layers and users subscribe to different layers according to their network conditions. If network conditions change, users may change the number of layers subscribed to accordingly. Such approaches suffer from a number of drawbacks. The frequent join-leave actions in a dynamic network are associated with significant process overhead at the end hosts in addition to the video stream re-synchronization problems at receiver. Also the number of layers at source is limited (usually 3-5 layers) and thus can not satisfy each user in a heterogeneous environment. We used the same network topology as in Fig. 3.25. Similar to Tan and Zakhor's paper [69], in the layered multicast scheme, the source sends out 4 layers with the base layer at 0.6 Mbps, and the other three enhancement layers are at 0.3 Mbps, 0.42 Mbps and 0.32 Mbps, respectively. The criteria of deciding the rate of each layer at the server is to satisfy as many users as it can. To generate the four layers, the source first encodes the video based on the highest bandwidth at 1.64 Mbps with FGA-FEC and then splits each encoded GOP into 4 layers at the designed rate. The receivers adapt to network conditions

by joining and leaving the appropriate layers according to available bandwidth. Meanwhile, our proposed FGA-FEC only sends out one encoded bitstream at 1.64 Mbps and adapts the encoded bitstream to suit diverse users' available bandwidth.



(a) Receiver 4

(b) Receiver 5

(c) Receiver 7

(d) Receiver 12

Figure 3.28: FGA-FEC vs. Layered multicast, quality delivered (in dB) at various receivers

In Fig. 3.28, we compare the delivered quality to users 4, 5, 7 and 12, using FGA-FEC and layered multicast, respectively. For user 4, the available bandwidth is 1.32 Mbps, the number of layers it can subscribe to is 3, which fits very well to the available bandwidth. In this case, both FGA-FEC and layered multicast receive the same video quality shown in Fig. 3.28(a). For user 5, the available bandwidth is 1.25 Mbps, the number of layers it can subscribe is two, which is up to 0.9 Mbps. For user 7, the available bandwidth is 1.38 Mbps, the number of layers it can subscribe is three which is up to 1.32 Mbps and very close to the available bandwidth of 1.38

Mbps. For user 12, the available bandwidth is 0.66 Mbps, and it can subscribe only to the base layer which is 0.6 Mbps. On the other hand, FGA-FEC adaptation is achieved by actively dropping packets and shortening descriptions. It matches the available bandwidth very well which has better performance in all three users. The adapted bitrate for user 5 is 1249.92 Kbps (available bandwidth is 1.25Mbps), for user 7 is 1379.52 Kbps (available bandwidth is 1.38 Mbps) and for user 12 is 659.52 Kbps (available bandwidth is 660 Kbps). The rate difference is mainly caused by adaptation precision which is one byte for each description, thus resulting in 64 bytes for a whole GOP. Since the number of layers in a multicast is limited, if more diverse users are involved in the transmission, there should be more users with mismatched video bitrate. For example, if a set of users have available bandwidth ranging from 200 Kbps to 2000 Kbps and there are four multicast layers available, only a small number of users can match their available bandwidth with available layers. FGA-FEC matches very well to the available bandwidth by adaptation.

Results in this section show FGA-FEC has much finer adaptation to diverse users and hence better quality compared to layered multicast.

3.4 Conclusions

In this chapter, we present a fine grain adaptive forward error correction coding approach for scalable video streaming. FGA-FEC encodes a scalable embedded video bitstream in such a way that both the video bitstream and the error control codewords can be easily and precisely adapted in a multidimensional way at intermediate overlay nodes to satisfy a diversity of users without complex transcoding. The adaptation at the intermediate overlay nodes is fine granular at the block level. Encoding once, the proposed scheme can adapt FEC codes by only adjusting and deleting packets instead of FEC decoding/recoding at the intermediate nodes. Simulations and experiments show that FGA-FEC can efficiently and precisely stream scalable video to multiple heterogeneous users. Future work will focus on cooperative adaptation between DSNs and extension to the wireless case with bit errors in the packets. Also, we will look at extension to the case where bandwidth is limited among the DSNs.

CHAPTER 4
Generalized FGA-FEC over Wireless Networks

In this chapter, we extend our proposed FGA-FEC coding scheme, a generalized MD-FEC method, to wireless networks. To protect the encoded scalable video bitstream over lossy channel and facilitate content adaptation at intermediate nodes, we use product codes based on BCH/CRC codes as row codes and RS codes as column codes. We propose a fast algorithm to optimize the product codes within several iterations from a near optimal point. Simulations show good performance in both content adaptation and protection.

4.1 Introduction

In Chapter 3, we proposed a fine grain adaptive forward error correction coding (FGA-FEC) scheme for overlay video streaming, that can encode a video to facilitate efficient and precise adaptation of the encoded bitstream to satisfy heterogeneous users without complex transcoding at intermediate overlay nodes. In this chapter, we generalize the proposed FGA-FEC scheme to wireless networks. In addition to congestion related packet losses, wireless networks have to deal with a time varying, error-prone, physical channel that in many instances is also severely bandwidth and computationally constrained. Therefore, bitstream protection and adaptation methods have to consider packet erasures due to congestion/route disruption as well as the following three unique wireless characteristics: (a) channel bit errors due to channel fading and noise, (b) large bandwidth fluctuation, and (c) intermediate node computational capability constraints.

For packet-erasure channels, efficient FEC is obtained by using systematic RS codes across packets. For fading channels, state-of-the-art performance is given by product channel codes, originally proposed by Sherwood and Zeger [121], to protect progressively compressed (embedded) and packetized images for noisy channels as shown in Fig. 4.1. Within packets, the product code uses the concatenation of a rate compatible punctured convolutional code (RCPC) and an error detecting cyclic

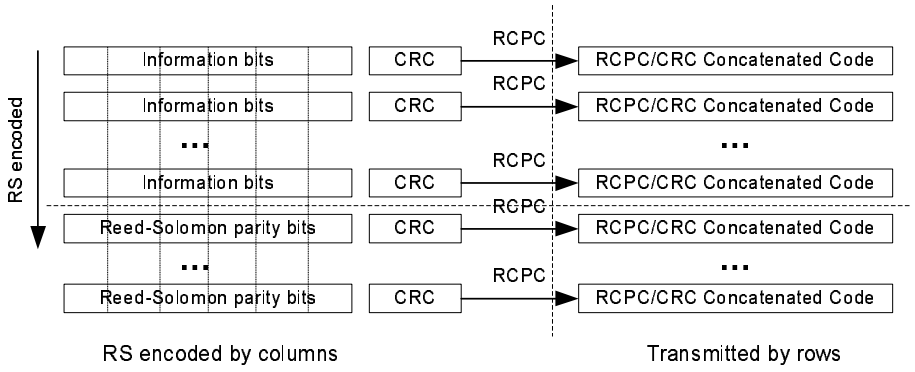


Figure 4.1: Schematic diagram of RCPC/CRC and RS product code

redundancy check (CRC) code as row codes. Across packets, RS codes are used as column codes. At the receiver, the row codes are first decoded. Decoding failure of the row codes is treated as erasures when decoding the column RS codes. Similar as [121], Sachs *et al* [122] introduced a multiple-description product code which aims at optimally generating multiple, equally-important wavelet image descriptions. Their error-correction codes are concatenated channel codes including row (outer) codes based on RCPC codes with CRC error detection, and source-channel column (inner) codes consisting of the scalable SPIHT image coder and an optimized array of unequal protection RS erasure-correction codes. Stankovic *et al* [119] proposed a low-memory linear-time iterative algorithm that jointly optimizes the RS and RCPC product codes. A good survey about optimized error protection of scalable image bitstreams over fading channels can be found in [123].

The above papers proved that product code is an efficient error protection method for end-to-end unicast scalable image transmission over fading channels. None of the papers considers the adaptation of product codes and the image data for multiple heterogeneous users. Compared to image coding, scalable video has more degrees of adaptation, and the users' requirements are also more diverse. Therefore, to protect scalable video for diverse users over error prone channels, we should consider not only the protection scheme, but also the feasibility of bitstream and error-control codes adaptation. In this chapter, we generalize the FGA-FEC scheme to incorporate product codes to address the additional problems in wireless network.

For a wireless network, after FGA-FEC encoding, we further encode each de-

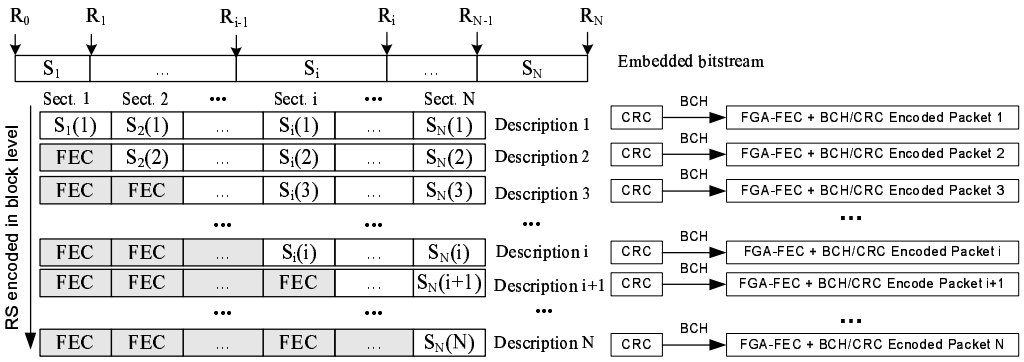


Figure 4.2: Generalized FGA-FEC with product codes

scription using a BCH code with CRC error detection to protect it from bit errors as shown in Fig. 4.2. We choose systematic BCH codes over RCPC codes based on the following reasons: (1) BCH codes are well known codes for binary data transmission, especially good for large block codes [114]. (2) Block codes are generally developed and analyzed through the use of algebraic/combinatorial techniques, while convolutional codes have been amenable almost solely to heuristic construction techniques. We need more control over the FEC. (3) Since we will do intermediate bitstream adaptation, we need to decode/re-code the bit level FEC, therefore we need to choose a code which can be fast decoded/re-encoded. We choose systematic codes instead of convolution codes. In the intermediate node, we first check the CRC code of each description. If CRC check passes, we can directly drop the parity bits without decoding the BCH codes, otherwise, a BCH decoding is triggered. On the other hand, if using RCPC codes, each RCPC codeword (description) should be decoded for adaptation. (4) BCH codes are cyclic codes and can be shortened, which fits very well to the bitstream adaptation.

In addition to protecting the scalable video bitstream using product codes, the generalized scheme also should keep the adaptivity feature of the original FGA-FEC scheme. In FGA-FEC, we encode one GOP of a scalable video bitstream into N descriptions (called single-cluster coding in this study). In the generalized scheme, we will encode one GOP into multiple clusters, with each cluster N descriptions (called multi-cluster coding). This results in a smaller packet size for each description. The encoding of one GOP into multiple clusters is motivated by the following

two reasons: (i) the maximum transmission unit size is limited in an error prone wireless channel, we want to fit one encoded description into one network packet; (ii) it facilitates adaptation at intermediate wireless nodes. For example, if each frame-rate layer is encoded as one cluster of descriptions, adaptation to lower frame-rates can be achieved by simply dropping related cluster from the encoded GOP.

Given the above generalized scheme, several important problems should be solved.

1. How should the FEC assignment be optimized, given both channel packet-loss rates and bit-error rate?

2. How should one GOP be encoded into multiple clusters of N descriptions, instead of just one cluster?

3. How should the bitstream be adapted at intermediate wireless nodes, given large channel bandwidth fluctuation and constrained computational capability at the network nodes.

4.2 Enhanced Link Layer Protocol

In the generalized FGA-FEC scheme, we protect the application payload (i.e., the video bitstream) using product codes, packets with errors should be passed to application for error correction. In a wireless network, the header part of each protocol layer is crucial, since intermediate nodes rely on this header information to forward a packet to its right destination. If the header has some errors in it, usually the whole packet is useless. To ensure correct delivery, we use a link-layer header FEC [102, 104] scheme to enhance the MAC/PHY layers (We will discuss this protocol in Chapters 5 and 6 in detail). A bit-level FEC is added to the packet header at link layer to protect this header from bit errors as shown in Fig. 4.3. A packet is dropped if the header CRC/FEC fails, otherwise, it is forwarded to the next node and ultimately up to the application. In this study, we use the enhanced MAC/PHY protocol to pass packets with errors to application layer.

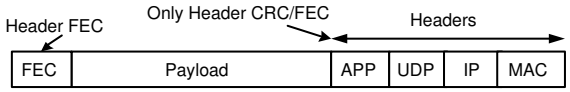


Figure 4.3: Enhanced MAC/PHY protocol using header FEC

4.3 Generalized FGA-FEC over Wireless Networks

Recall the generalized FGA-FEC encoding scheme as shown in Fig. 4.2, at receiver side, the BCH decoder first tries to correct bit errors within each description. If BCH decoding fails in a certain description (based on CRC check result, we assume that the CRC can always detect the errors within one description), that description is dropped as an erasure. The RS decoder then tries to decode the column codes with only erasures.

FGA-FEC can encode and adapt the product codes based on both channel conditions and user video preference, as well as user predefined adaptation order. A user can chose to adapt downward quality, frame rate and resolution in any particular order. Suppose a user's video preference is to view a video at L_t temporal layer, L_s spatial layer and PSNR$\geq \gamma'$ dB, and the user's minimum tolerable bitstream is at $L_{t\,\min}$ temporal layer, $L_{s\,\min}$ spatial layer and PSNR$\geq \gamma$ dB, $\gamma' \geq \gamma$. Therefore, the user's video request ranges from $\{L_t, L_s, \gamma'\}$ to $\{L_{t\,\min}, L_{s\,\min}, \gamma\}$. Along the user's adaptation order, the server or intermediate nodes need to deliver the best possible video for this user within its requested bitstream range in response to available bandwidth. First, we need to find the optimal product code assignment for a given scalable video bitstream over a certain BER channel under a bitrate constraint. We start from encoding one GOP into one cluster of N descriptions, since this is the starting point for multiple clusters assignment.

4.3.1 Single-cluster Coding

The optimization problem is to find a concatenated column RS code assignment c_c and row BCH code assignment c_r from a set of RS codes C_{RS} and BCH codes C_{BCH}, such that the end-to-end distortion is minimized and the corresponding PSNR$\geq \gamma$ dB.

$$c_c, c_r = \underset{c_c \in C_{RS},\ c_r \in C_{BCH}}{\operatorname{argmin}} E[D|C_{RS}, C_{BCH}, C_{CRC}], \tag{4.1}$$

subject to:

$$R_s + R_{RS} + R_{CRC} + R_{BCH} \leq B,$$

where C_{CRC} is the CRC code set, R_s is the source rate, R_{RS} is the rate allocated to RS parity bits, and R_{CRC} (R_{BCH}) are the rates allocated to CRC (BCH) check bits. B denotes the maximum available channel bitrate. Since CRC codes are only used for error detection, we use a fixed 32 bit CRC code in this study and hence, R_{CRC} is constant.

For any integer $m \geq 3$ and $t < 2^{m-1}$, there exists a primitive BCH code with the following parameters:

$$n = 2^m - 1$$
$$n - k \leq mt$$
$$d_{min} \geq 2t + 1$$

This code can correct t or fewer random errors over a span of $2^m - 1$ bit positions. The code is a t-error-correcting BCH code and represented as BCH(n, k, t). Due to the discrete nature of BCH codes, given n, we know that $n \geq k \geq n - mt$. Therefore, we can solve (4.1) by a simple search algorithm as following: Given one BCH(n, k, t) code assignment, find the optimal RS code assignment under bitrate constraint $R_s + R_{RS} \leq B - R_{CRC} - R_{BCH}$, until exhaust all possible BCH codes at C_{BCH}. The one with minimum end-to-end mean distortion is the solution of (4.1). We call this method exhaustive search. Later on, we will describe a more efficient search algorithm that starts from a near optimal point and can find the optimal product codes assignment within several iterations. Here, one iteration is defined as one RS code assignment optimization.

Given a BCH (n, k, t) codeword, number of bit errors larger than t in the codeword cannot be corrected, using a BSC channel, the probability of not correctly decoding the codeword is

$$P_{BCH}(E) = \sum_{j=t+1}^{n} \begin{pmatrix} n \\ j \end{pmatrix} p_b^j (1-p_b)^{n-j}, \tag{4.2}$$

where p_b is the channel bit error rate. Decoding failures in the row codes are treated as erasures when decoding the column RS codes. Since the column codes typically only need to correct erasures, the computational complexity is reduced, and also twice as many lost rows can be recovered compared to a decoder without error detection on the rows. Given bit-error rate p_b, and the probability of a packet being dropped due to congestion/route disruption is p_{drop}. The probability of a packet erasure p after BCH decoding can then be approximated as:

$$p = p_{\text{drop}} + (1 - p_{\text{drop}}) \times P_{BCH}(E), \tag{4.3}$$

Now we are encoding one GOP into a single cluster of N descriptions, after assigning a BCH code and a CRC code, the available bandwidth for RS codes and video data is updated to $B - R_{CRC} - R_{BCH}$. We need to optimize the assignment of column codes under this rate constraint. The goal is to find the optimal bitrate partition $R = \{R_1, R_2, ..., R_N\}$ in Fig. 4.2, which minimizes the end-to-end mean distortion $E[D(R)]$, and the corresponding PSNR$\geq \gamma$ dB.

$$E[D(R)] = \sum_{i=0}^{N} q_i D(R_i), \tag{4.4}$$

subject to:

$$\begin{cases} 0 \leq R_1 \leq R_2 \leq ... \leq R_N \\ R_{\text{total}} \leq B \\ R_i - R_{i-1} = r_i \times i, \qquad r_i \geq 0, \quad \forall\ i \in [1, N] \end{cases}$$

where r_i is the rate of each subsection at section i ($i \in [1, N]$) of the bitstream (please refer to Chapter 3 for definition of sections and subsections). Let q_i be the probability that any i out of N packets are successfully delivered, given a packet-loss

probability p and assuming independent losses, the q_i can be calculated as:

$$q_i = \begin{pmatrix} N \\ i \end{pmatrix} (1-p)^i p^{N-i}, \tag{4.5}$$

where p can be calculated using (4.3). The bitrate R_{total} is the total bandwidth (bitrate) available for both FEC and video data and can be calculated as:

$$\begin{aligned} R_{\text{total}} &= R_s + R_{RS} + R_{CRC} + R_{BCH} \\ &= \frac{R_1}{1}N + \frac{R_2 - R_1}{2}N + \cdots + \frac{R_N - R_{N-1}}{N}N + R_{CRC} + R_{BCH} \\ &= \sum_{i=1}^{N} \frac{N}{i(i+1)} R_i + R_{CRC} + R_{BCH} \\ &= \sum_{i=1}^{N} \alpha_i R_i + R_{CRC} + R_{BCH}, \end{aligned} \tag{4.6}$$

where $\alpha_i = \frac{N}{i(i+1)}$ for $i = 1, 2, \cdots, N-1$; and $\alpha_N = 1$.

Solving (4.4) is a constrained optimization problem. To find the optimal solution, we can use the Lagrange multiplier method and construct a function:

$$F(R_1, \cdots, R_2, \lambda) = \sum_{i=0}^{N} q_i D(R_i) + \lambda(\sum_{i=0}^{N} \alpha_i R_i + R_{CRC} + R_{BCH} - B), \tag{4.7}$$

The BCH(n, k, t) code and CRC code are given at a certain optimization step, then R_{CRC} and R_{BCH} are constant. Taking the partial derivative of (4.7) with respect to R_i, $i = 0, 1, \cdots, N$ and λ, setting them to 0. The optimal assignment is the solution of the following:

$$\begin{cases} \frac{dD(R_1)}{dR_1} = -\frac{\alpha_1}{q_1}\lambda \\ \cdots \\ \frac{dD(R_N)}{dR_N} = -\frac{\alpha_N}{q_N}\lambda \\ R_{\text{total}} = \sum_{i=0}^{N} \alpha_i R_i + R_{CRC} + R_{BCH} = B \end{cases}$$

Given a value λ, $\frac{dD(R_i)}{dR_i} = -\frac{\alpha_i}{q_i}\lambda$ is essentially corresponding to one point at

the bitstream $D(R)$ curve with slope equal to $-\frac{\alpha_i}{q_i}\lambda$ and the bitrate is R_i. To match the total budget B, we can use a bisection search (Algorithm 3) to find the smallest λ and the corresponding rate break points (see Fig. 4.2 top). Please note that at every adaptation level, we need to use different $D(R)$ curves for the RS codes optimization, an example is shown at Fig. 4.4, where we show the $D(R)$ curves in different frame rate and resolution of *Foreman* sequence.

Algorithm 3: Bisection search algorithm

1. Initialize, set λ to a high value;
2. Calculate $-\frac{\alpha_i}{q_i}\lambda$ for all $i \in [1, N]$, $-\frac{\alpha_i}{q_i}\lambda$ is the slope of the $D(R)$ curve at a certain point, this point corresponds to a rate R_i at the $D(R)$ curve;
3. Calculate R_{total} using equation (4.6);
4. Test if $R_{\text{total}} = B$, If so, solution found, **stop**, $R = \{R_1, R_2, \cdots, R_N\}$. Otherwise, go to two branches: If $R_{\text{total}} > B$, total rate is too high, go to step 5, If $B > R_{\text{total}}$, total rate is too low, go to step 6;
5. Bisection search, reduce λ, set $\lambda = \frac{\lambda}{2}$, go to step 2;
6. Reverse search, increase λ, set $\lambda = \lambda + (\lambda_{pre} - \lambda)/2$, where λ_{pre} is the λ value at previous step, go to step 2.

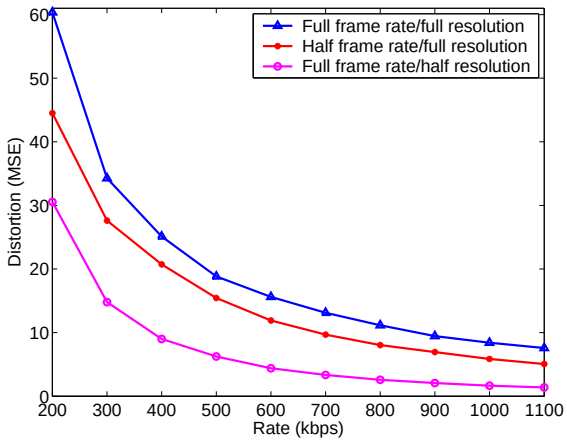


Figure 4.4: $D(R)$ curves at various adaptation levels.

Optimal column code assignment at a certain given BCH code and CRC code is a constrained optimization problem and can be solved by using Lagrange multi-

plier method as described above. The optimal product code could be achieved by exhaustively searching over all possible BCH codes along the adaptation order. The total number of iterations is the same as choosing all possible t, which is $t \in [0, n/m]$. The computational burden is heavy if exhaustive search is performed for each GOP. We propose a fast search algorithm which can find the optimal product codes assignment within several iterations from a near optimal point.

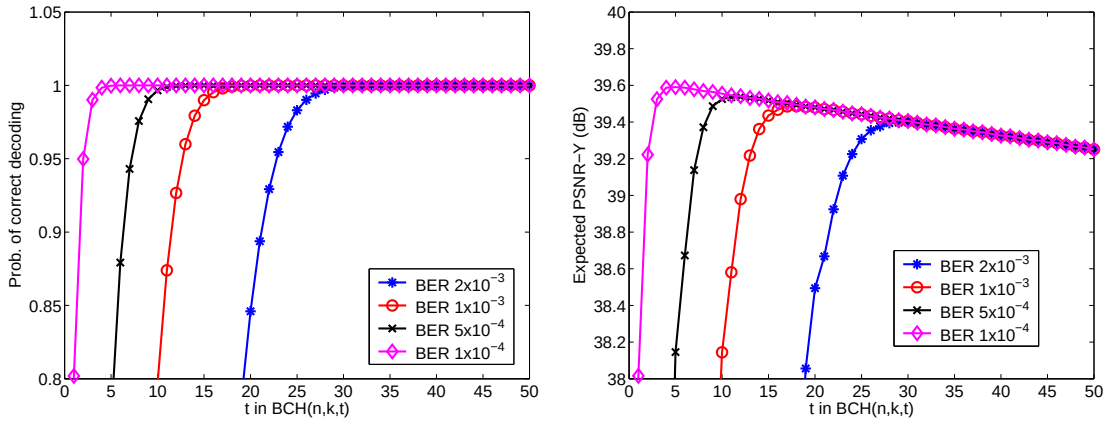
4.3.2 Fast Search Algorithm

From 4.3, we know that BCH decoding error contributes to packet loss probability. At a certain BER, stronger row codes would result in a lower probability of decoding error, thus reduce the probability of packet erasure. On the other hand, allocating more bandwidth to BCH codes would result in less bandwidth allocated to the video source and RS codes, hence higher distortion. Therefore, we can find the optimal point by leveraging these two factors.

We first did tests via exhaustive search over videos *Foreman* (CIF, 18 GOPs), *Mobile* (SIF, 8 GOPs) and *Football* (SIF, 7 GOPs) at various BER, available bandwidth, as well as number of descriptions. Fig. 4.5 shows an example in one of these tests. The task is to protect MC-EZBC encoded *Foreman* CIF sequence, GOP #7. Here, $N = 64$, $B = 980$ Kbps, $p_{\text{drop}} = 0.05$, $p_b = 2 \times 10^{-3}$, 1×10^{-3}, 5×10^{-4} and 1×10^{-4}, respectively. The set of BCH codes are BCH(n, k, t), where $n = 8191$, $m = 13$, $k = n - mt$, fixed 32 bits CRC code, therefore t can vary from 0 to 623.

In the exhaustive search, we progressively set t in BCH codes from 0 to 623 (exhaust all possibilities), update the available bandwidth and optimize the column RS codes at each BCH assignment. Finally, find the minimum expected distortion among all iterations. Fig. 4.5(a) shows the probability of successfully decoding these BCH(n, k, t) codes at given BERs. Fig. 4.5(b) shows the zoomed corresponding optimized PSNR vs. t. Table 4.1 shows the optimization results of this test.

Our key observation is that the points near the knee of Fig. 4.5(a) are near optimal points in Fig. 4.5(b). Therefore we can pick up a starting point t from these knee points and locally search to find the optimal solution. Since the expected distortion $E[D]$ curve is concave around the optimal point, we can fist test three



(a) (b)

Figure 4.5: (a) probability of successful BCH(n, k, t) decoding at various channel BER vs. t; (b) Average PSNR of video vs. t

BER	$E[D]$(PSNR, dB)	BCH(n, k, t)
0	39.63	none
1×10^{-4}	39.59	BCH(8191,8125,5)
5×10^{-4}	39.54	BCH(8191,8034,12)
1×10^{-3}	39.49	BCH(8191,7956,18)
2×10^{-3}	39.40	BCH(8191,7800,30)

Table 4.1: The results of optimal assignment at different BER

points $(t-1, t, t+1)$, find the search direction of t. After that, we progressively allocate bandwidth to BCH codes along the search direction, and then optimize the column RS codes at each BCH code assignment, until find the optimal point.

We use a threshold method to choose the starting point. We pick a value of the threshold ε, and test the probability of correctly decoding a BCH(n, k, t) code, $P_{BCH}(C)$, at a certain BER with ε, where $P_{BCH}(C) = 1 - P_{BCH}(E)$. The smallest point t with $P_{BCH}(C) > \varepsilon$ is the starting point. Obviously, different threshold ε corresponds to different optimization performance in terms of number of iterations to reach the optimal point. Simulations show that $\varepsilon = 0.999$ is a good starting point. Experiments show that good convergence is obtained with just three to five iterations on average. Formula (4.2) used in Fig. 4.5(a) can be stored in a small table, which further reduces the computational burden and fasten the optimization process. Algorithm 4 summaries the product code optimization method.

Algorithm 4: Product code optimal assignment

Input : B, p_b, p_{drop}, N
Output: c_r, c_c

1 Pick starting point t, such that $P_{BCH}(C) > \varepsilon$;
2 Assign BCH(n,k,t) code;
3 Calculate p as (4.3);
4 Optimize RS codes $t-1, t, t+1$, calculate $E[D]_{t-1}$, $E[D]_t$, and $E[D]_{t+1}$;
5 If $E[D]_t \leq \min(E[D]_{t-1}, E[D]_{t+1})$, go to Step 9;
6 If $E[D]_{t-1} \leq \min(E[D]_t, E[D]_{t+1})$, search lower t, go to Step 8;
7 If $E[D]_{t+1} \leq \min(E[D]_{t-1}, E[D]_t)$, search higher t;
8 Iterate on t a few steps;
9 If PSNR$\geq \gamma$, solution found, **Stop**, otherwise, move down one adaptation level following user's adaptation order.
10 If adaptation level exhausted, **Stop**, no video is sent. Otherwise, go to Step 4;

While full Lagrange-based optimization is performed at the server, only FGA-FEC adaptation consisting of shortening and/or dropping packets is done at intermediate nodes [98, 99].

4.3.3 Multi-cluster Coding

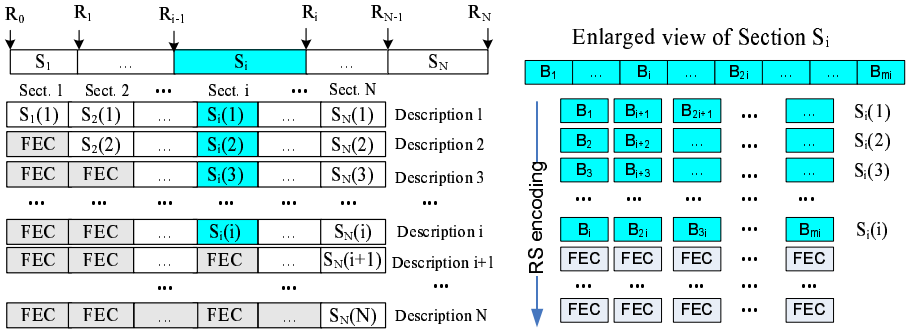


Figure 4.6: FGA-FEC encoding method, one GOP of bitstream is encoded to N descriptions, RS codes are applied across descriptions vertically at block level.

In wired network, FGA-FEC encodes one GOP into single cluster of N descriptions as shown in Fig. 4.6. Given an available bit budget Ω for one GOP, the size of each description can be calculated as Ω/N. In a wireless network, the transmission packet size is usually limited. Hence, one Maximum Transmission Unit (MTU) may

not be large enough to transmit a FGA-FEC encoded description. Therefore, we should use smaller description size. One solution to achieve this is to use a larger N. The problem with this approach is that the computational burden would increase (due to long codewords) and the adaptation precision becomes coarse as N grows, since we must adapt out more data in one column of blocks. Another solution, we could encode one GOP to multiple clusters of descriptions, also resulting in smaller packet size for each description. Since each column in Fig. 4.6 is randomly accessible, an easy way to do this is to assign FEC over an entire GOP as FGA-FEC does for a wired network, and then split each description into several new descriptions. For example, each description can be split into two smaller descriptions by collecting odd column blocks and even column blocks in Fig. 4.6, respectively. We will stick to the latter solution and encode one GOP into multiple clusters of descriptions, with each cluster N descriptions.

Scalable data sources are generally composed of nested elements. The compression of these elements imposes dependencies so that a source element might not be decodable without first correctly decoding other elements. Usually, a scalable video bitstream has three basic types of scalability: temporal (frame-rate) scalability, spatial (resolution) scalability, and SNR (quality) scalability. This kind of bitstream can be encoded in such a way that the subsets corresponding to lower frame-rate /resolution /quality of the video are embedded in bitstreams corresponding to higher frame-rate /resolution /quality. Different sub-bitstreams can be extracted in a simple manner without transcoding, to readily accommodate a variety of users considering their computing power, connection bandwidth. Therefore, in order to optimally encode one GOP of a scalable bitstream into multiple clusters of descriptions, we first need to explore the dependency of the encoded bitstream and to figure out how we should split the encoded GOP. We use MC-EZBC encoded bitstream as example to show the bitstream dependency.

Fig. 4.7 shows the bitstream dependency of an embedded scalable bitstream where each node denotes a sub-bitstream at a certain spatial and temporal layer. The subbands from Y, U and V belonging to that scale are progressively encoded from the most significant bit (MSB) towards the least significant bit (LSB) and

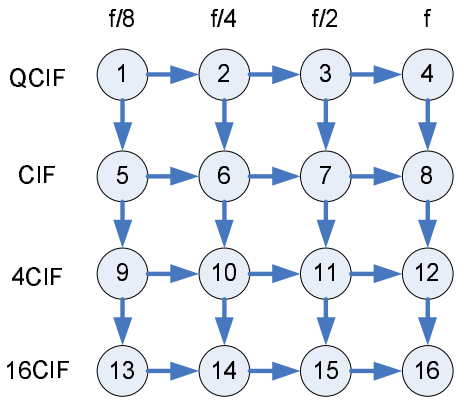


Figure 4.7: Bitstream dependency of an embedded bitstream, where each node denotes a piece of bitstream at certain frame-rate and resolution, arrows denote the dependency, for example, node 2 depends on node 1

each bitplane pass n is further divided into several sub-bitplane passes. These sub-bitstreams are each coded as independent arithmetic codewords and are addressable in the compressed file. We can choose to transmit any spatiotemporal resolution sub-bitstreams. Each pass has an entry in the bit allocation table showing its size. This is intended to access each pass in the compressed file, randomly. Since the spatiotemporal layers are independently encoded, the compressed data corresponding to the various layers can be arranged in different configurations to create a rich set of progression orders to serve different applications. The bitstream shown at Fig. 4.7 can be reorganized as Fig. 4.8.

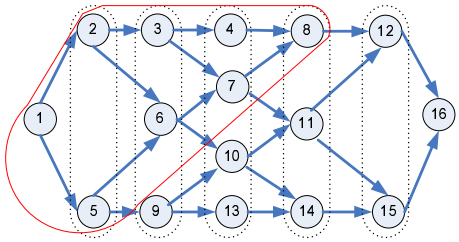


Figure 4.8: Reorganized bitstream dependency, each dash lined group depends on its parent bitstream from left, the dependency is in groups

Fig. 4.8 represents the reorganized dependency of bitstreams in groups. Nodes in one dash-lined ellipse is one group. Each group depends on its parent group from the left. For example, node 1 is the root node. Nodes 3, 6, 9 comprise one group and depends on nodes 2, and 5, also one group. Each low frame-rate and resolution

bitstream can be extract from the graph. For an instance, we can directly extract the full frame-rate CIF bitstream from the solid line circled area as Fig. 4.9. Also the quality layers within each node can be stopped at any point of the sub-bitstream from MSB to LSB.

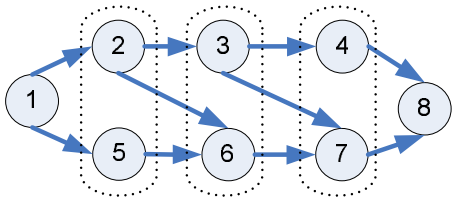


Figure 4.9: Sample of full frame rate and CIF bitstream can be extracted from the reorganized bitstreams

Based on the above dependency analysis, we know that certain elements can not be correctly decoded without first successfully decoding some earlier elements that they depend on. Therefore, we should add more protection to the more important elements. Further, the spatiotemporal layers are independently encoded bitplane by bitplane and can be accessed independently, we can arrange the sub-bitstreams in different progression orders to serve diverse users. In the scenario of multiple clusters of description encoding, elements are arranged into multiple clusters. Considering the randomly accessible characteristics of an embedded bitstream element, it is possible to process one GOP as an entity and generate N descriptions without the limit to descriptions size (We give this type of description a term "*computation description*", since they are only used for optimization and are not transmitted, until split into new smaller descriptions), then re-order and split the FGA-FEC encoded bitstream into multiple clusters by splitting each computation description into multiple new descriptions. In the receiver, these clusters are concatenated together to reconstruct the computation descriptions and further the GOP. Column RS code optimization is similar as FGA-FEC at wired network, the problem we need to solve in this part is how to optimally assign the row BCH codes for each cluster so that the product codes assignment for this GOP is optimal.

Suppose we want to encode one GOP into multiple clusters of descriptions, say m clusters, with N descriptions in each cluster. The size of a description in cluster j is n_j in bits for all $j \in [1, m]$, and this parameter is known before encoding. The

BCH code applied to cluster j is $BCH(n_j, k_j, t_j)$. The task is to find concatenated column RS codes assignment $c_c(j)$ and row BCH codes assignment $c_r(j)$ for cluster $j,\ j \in [1, m]$, from a set of RS codes C_{RS} and BCH codes C_{BCH}, such that the end-to-end distortion of the encoded GOP is minimized.

$$c_c(j), c_r(j) = \underset{c_c(j)\in C_{RS},\ c_r(j)\in C_{BCH}}{\operatorname{argmin}} E[D|C_{RS}, C_{BCH}, C_{CRC}],\ \forall j \in [1, m] \tag{4.8}$$

subject to the total rate constraint:

$$\textstyle\sum_{j=1}^{m}(R_s(j) + R_{RS}(j) + R_{CRC}(j) + R_{BCH}(j)) \leq B,$$

where, for the jth cluster, $R_s(j)$ is the source rate, $R_{RS}(j)$ is the rate allocated to RS parity bits, and $R_{CRC}(j)$ ($R_{BCH}(j)$) are the rates allocated to CRC (BCH) check bits.

The total rate constraint in (4.8) can be re-written to:

$$\sum_{j=1}^{m}(R_s(j) + R_{RS}(j)) \leq B - \sum_{j=1}^{m}(R_{CRC}(j) + R_{BCH}(j)) \tag{4.9}$$

Given FGA-FEC encoding method as shown in Fig. 4.6, each vertical block column is independently accessible, the left side of (4.9) can be further re-written to:

$$\sum_{j=1}^{m}(R_s(j) + R_{RS}(j)) = \sum_{j=1}^{m} R_s(j) + \sum_{j=1}^{m} R_{RS}(j) = R_s + R_{RS} \tag{4.10}$$

Based on (4.9) and (4.10), as well as the FGA-FEC encoding method in block level, we can first optimization FEC protection based on the entire GOP without the limit to description size as in Fig. 4.6 and then split the encoded GOP into multiple clusters with $\sum_{j=1}^{m} k_j = k$, where k is the size of the computation description in Fig. 4.6, and the BCH code applied to cluster j is $BCH(n_j, k_j, t_j)$. The splitting would not affect the optimality, since we do not change any of the FEC assignments (both BCH codes and RS codes). The criteria of the splitting is to facilitate intermediate overlay adaptation. Fig.4.10 shows an example of splitting one encoded GOP into two clusters of descriptions, each clusters has N descriptions, $k = k1 + k2$. This

splitting facilitates frame rate adaptation, one cluster can be directly dropped at intermediate nodes for half frame rate.

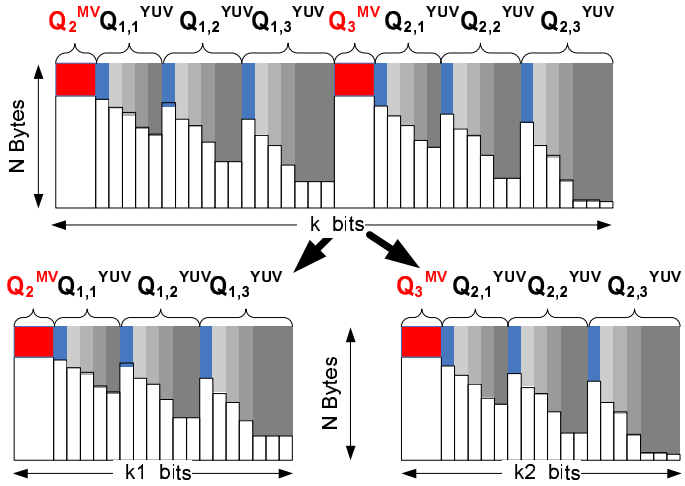


Figure 4.10: An example of splitting one encoded GOP into two clusters of descriptions, blank blocks contains FEC, each description in the two clusters is coded with BCH codes horizontally.

Similar as single-cluster optimization, in multiple-cluster optimization, we progressively allocate bandwidth to BCH codes for each cluster, and then optimize the RS code protection, until exhaust all possible BCH codes at C_{BCH}. The one with minimum distortion is the solution to (4.8). One new problem we need to solve in multiple-cluster optimization is how we should distribute the BCH bandwidth to each cluster, given the total allocated bandwidth to BCH codes for all clusters. If using the splitting method to generate multiple clusters, one new description error, could cause the computation description useless. The probability of one computation description being successfully transmitted to the destination is determined by the successfully transmission of all split descriptions of the computation description.

$$P(C) = \prod_{j=1}^{m}(1-p_{\text{drop}})P_{BCH}(C)_j = (1-p_{\text{drop}})^m \prod_{j=1}^{m}\sum_{i=0}^{t_j} p_b^i(1-p_b)^{n_j-i} \tag{4.11}$$

The probability of one computation description error

$$p = 1 - P(C) \tag{4.12}$$

This p is used to optimize the RS codes assignment as packet loss probability.

Algorithm 5: Multiple-cluster optimization

Input : B, p_b, p_{drop}, N, m, n_j, $\forall j \in [1, m]$
Output: $c_r(j)$, $c_c(j)$, $\forall j \in [1, m]$

for $(\Phi = 0; \Phi \leq \Sigma_{j=1}^{m} n_j;\ \Phi++)$ **do**
  find maximum $P(C)$, subject to $\sum_{j=1}^{m} t_j = \Phi$;
  Assign CRC code for each cluster;
  Assign $BCH(n_j, k_j, t_j)$ codes for cluster j, $\forall j \in [1, m]$;
  Calculate p as (4.12);
  RS code optimization;
  Calculate $E[D(R)]_{\Phi}$ for this iteration;
end
Find $c_r(j)$, $c_c(j)$, $\forall j \in [1, m]$ which corresponds minimum $E[D(R)]_{\Phi}$;
Solution Found;

Algorithm 5 summaries the procedure of multiple-cluster optimization. We progressively allocate bandwidth to BCH codes. Given the total bit budget allocated to BCH codes is Φ bits for each computation description, the bit budget is distributed to the m clusters, such that $\sum_{j=1}^{m} t_j = \Phi$. We can find the maximum $P(C)$, subject to $\sum_{j=1}^{m} t_j = \Phi$ by a simple search algorithm. Then the probability of a computation description loss p is calculated based on BCH assignment to each cluster. After updating R_{total}, RS codes assignment is optimized as wired FGA-FEC. This procedure is repeated until exhausting all the BCH codes in C_{BCH}, the product codes assignment corresponds to the minimum mean end-to-end distortion of this GOP is the solution. After optimization, computation descriptions are split to new descriptions with the description size of jth cluster k_j, then jth cluster is encoded with $BCH(n_j, k_j, t_j)$.

Fig. 4.11 summaries the generalized FGA-FEC scheme for wireless networks. The FEC protection is first optimized based on the network conditions (packet loss rate, BER and available bandwidth), then the RS codes are applied to the blocks vertically as shown in Fig. 4.6. The RS encoded GOP is split into multiple clusters of descriptions and each cluster is encoded with appropriate BCH codes over each description. Inside the network, the encoded GOP can be adapted based on user video requirement. At the receiver side, BCH decoder first tries to correct bit errors

inside each description and then the entire GOP is concatenated together. RS decoder decodes all the RS codes to reconstruct the original bitstream.

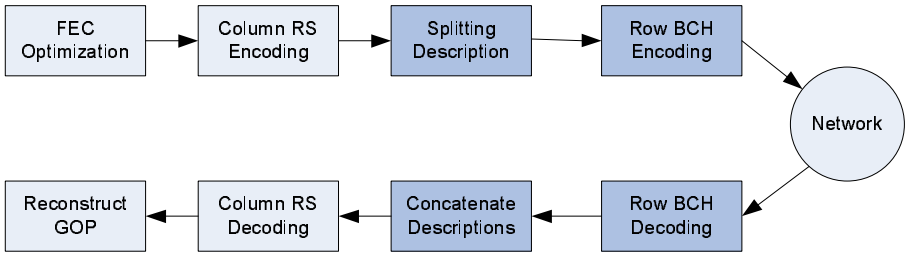


Figure 4.11: Generalized FGA-FEC for wireless network, the shadowed blocks are unique for wireless network

We tested Algorithm 5 in the following experiment. The task is to encode MC-EZBC coded *Foreman* GOP #7 into two and three clusters, respectively. Each computation description is split into two or three descriptions in two clusters coding or three clusters coding, respectively. The number of descriptions is 64 at each cluster, packet drop probability is set to $p_{\text{drop}} = 0.05$. Channel BER (p_b) is set to 2×10^{-3}, 1×10^{-3}, 5×10^{-4} and 1×10^{-4}, respectively. The available bandwidth is 985 Kbps.

Table 4.2 shows the expected distortion (PSNR in dB) of the optimal product codes assignment at different BER for encoding one GOP into two clusters. Interestingly, we found that the maximum $P(C)$ is always achieved when the allocated bandwidth to BCH codes is evenly distributed to each cluster, i.e descriptions in different clusters use the same BCH code. Therefore, we can use the fast optimization algorithm to optimize product codes for multiple clusters. BCH code used in each cluster is $BCH(\frac{n}{m}, \frac{k}{m}, \frac{t}{m})$, where m is the number of clusters in one GOP, n, k, t is the size in computation description. Fig. 4.12 shows the procedure of the optimization.

Table 4.3 shows the expected distortion (PSNR in dB) of the optimal product codes assignment at different BER for encoding one GOP into three clusters, Fig. 4.12 shows the procedure of the optimization.

Table 4.4 shows the encoding quality losses in dB by comparing multi-cluster coding with single cluster coding. The reason is because the longer BCH code at single cluster coding is always better than shorter code with the same amount of

BER	E[D] (PSNR, dB)	BCH(n, k, t)
0	39.630	none
1×10^{-4}	39.572	BCH(4095,4046,4)
5×10^{-4}	39.513	BCH(4095,3998,8)
1×10^{-3}	39.456	BCH(4095,3962,11)
2×10^{-3}	39.363	BCH(4095,3878,18)

Table 4.2: The results of optimal product codes assignment at different BER (encoding one GOP into two clusters)

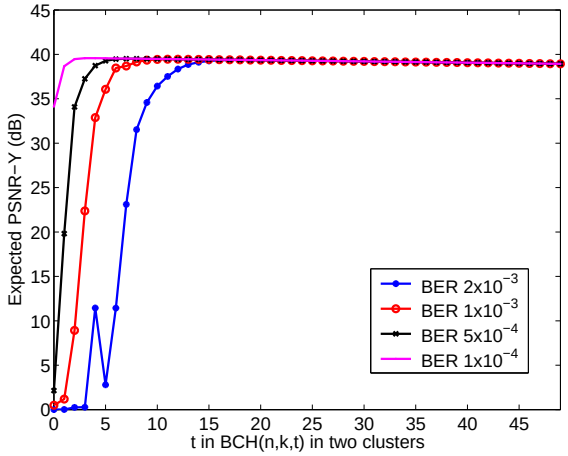


Figure 4.12: The optimization procedure of Algorithm 5 for two clusters assignment. BCH(n, k, t) vs. expected distortion (PSNR) at each BCH code iteration.

protection at multiple clusters coding. Results show that the encoding loss in quality is negligible, but with a much easier adaptation in multi-cluster coding.

4.4 Simulations

We performed simulations and experiments using test sequences: *Foreman* CIF, 300 frames, *Mobile* SIF, 128 frames and *Football* 112 frames. All sequences are 30 fps, 16 frames/GOP. The scalable source coder is MC-EZBC. Reed-Solomon codes are employed as column codes and BCH codes are applied to each description as row codes. The number of descriptions encoded in one GOP is 64. Each simulation is run at least ten times, and we present only averages for statistically meaningful results.

BER	E[D] (PSNR, dB)	BCH(n, k, t)
0	39.630	none
1×10^{-4}	39.564	BCH(2730,2694,3)
5×10^{-4}	39.489	BCH(2730,2658,6)
1×10^{-3}	39.428	BCH(2730,2622,9)
2×10^{-3}	39.321	BCH(2730,2562,14)

Table 4.3: The results of optimal product codes assignment at different BER (encoding one GOP into three clusters)

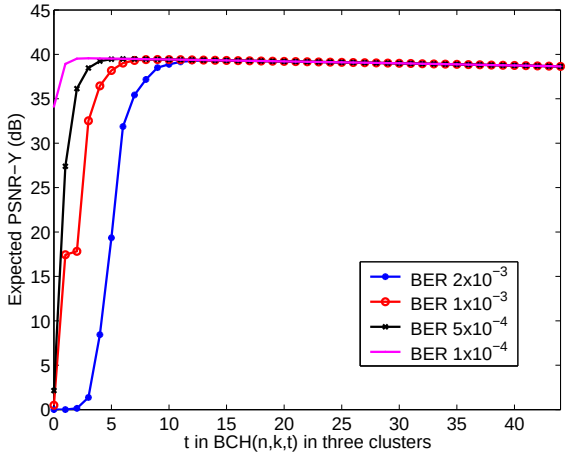


Figure 4.13: The optimization procedure of Algorithm 5 in three clusters assignment. BCH(n, k, t) vs. expected distortion (PSNR) at each BCH code iteration.

4.4.1 Optimization Performance

We propose fast optimization algorithm to assign product codes. We want to know how fast it can reach the optimal solution. Here, we count the number of iterations needed to optimize the product code of one GOP. Again, we start from single cluster coding.

4.4.1.1 Single Cluster Coding

The task is to encode all three sequences GOP by GOP, the available bandwidth is $B = 1$ Mbps, packet drop probability $p_{\text{drop}} = 0.05$. BER (p_b) is set to 1×10^{-4}, 1×10^{-5}, 1×10^{-6}, respectively, fixed 32 bits CRC code. We use the threshold $\varepsilon = 0.999$ for picking up the optimization starting point.

Table 4.5 compare the average number of iterations needed to reach the optimal

BER	Two clusters (dB)	Three clusters (dB)
0	0	0
1×10^{-4}	-0.018	-0.026
5×10^{-4}	-0.025	-0.049
1×10^{-3}	-0.031	-0.059
2×10^{-3}	-0.035	-0.077

Table 4.4: The encoding quality losses (PSNR loss in dB) of multi-cluster coding with single cluster coding

BER	Foreman	Mobile	Football
1×10^{-3}	3.94	3.88	4.0
1×10^{-4}	4.56	3.38	3.0
1×10^{-5}	4.11	3.25	3.0

Table 4.5: The average number of optimization iterations to reach optimal point at different BER

point of the three sequences. And Table 4.6 lists the optimization results, which exactly fit the result of an exhaustive search.

BER	Foreman (dB)	Football (dB)	Mobile (dB)
1×10^{-3}	38.43	27.55	28.72
1×10^{-4}	38.52	27.66	28.84
1×10^{-5}	38.53	27.68	28.86

Table 4.6: The optimal expected distortion (PSNR in dB) of different sequences at different BER

Table 4.7 compare the average number of iterations needed to reach the optimal point at different threshold ε. Based on the results, we found $\varepsilon = 0.999$ is a good threshold to choose the appropriate starting point for product codes optimization.

The simulations show that the proposed fast optimization scheme can quickly converge to optimal point within several iterations, which is very efficient.

4.4.1.2 Multi-cluster Coding

Now, we test the optimization algorithm by encode one GOP into multiple clusters. The sequences and the network setups are the same as single cluster coding,

BER	Foreman			Football			Mobile		
	ε_1	ε_2	ε_3	ε_1	ε_2	ε_3	ε_1	ε_2	ε_3
1×10^{-3}	5.19	3.94	5.83	5.0	4.0	6.0	4.75	3.88	6.5
1×10^{-4}	5.56	4.56	4.61	4.0	3.0	4.0	4.5	3.38	4.25
1×10^{-5}	4.72	4.11	4.11	4.0	3.0	3.0	4.25	3.25	3.25

Table 4.7: The average number of optimization iterations to reach optimal point at different BER and different threshold, $\varepsilon_1 = 0.99$,$\varepsilon_2 = 0.999$ and $\varepsilon_3 = 0.9999$

the only difference is the number of clusters encoded. Compare with single cluster coding, multi-cluster coding first need to distribute BCH codes to different clusters. Table 4.8 compares the average number of iterations needed to reach the optimal point at different threshold at 2 clusters case.

BER	Foreman			Football			Mobile		
	ε_1	ε_2	ε_3	ε_1	ε_2	ε_3	ε_1	ε_2	ε_3
1×10^{-3}	4.67	3.39	5.44	4.14	3.43	5.43	4.13	5.5	6.5
1×10^{-4}	4.94	3.94	3.06	4.06	4.0	4.0	5.38	4.25	3.38
1×10^{-5}	4.0	3.94	3.06	3.0	4.0	4.0	4.38	4.38	3.38

Table 4.8: Two clusters coding. The average number of optimization iterations needed to reach optimal point at different BER and different threshold, where $\varepsilon_1 = 0.99$,$\varepsilon_2 = 0.999$ and $\varepsilon_3 = 0.9999$

Table 4.9 lists the average number of iterations needed to reach the optimal point at different threshold at 3 cluster case.

BER	Foreman			Football			Mobile		
	ε_1	ε_2	ε_3	ε_1	ε_2	ε_3	ε_1	ε_2	ε_3
1×10^{-3}	5.06	3.22	4.83	4.71	3.29	5.14	4.63	3.63	5.88
1×10^{-4}	4.06	3.28	4.11	4.57	3.86	4.0	4.25	3.38	4.13
1×10^{-5}	3.0	3.0	4.00	3.0	3.0	4.0	3.0	3.0	4.0

Table 4.9: Three clusters coding. The average number of optimization iterations needed to reach optimal point at different BER and different threshold, where $\varepsilon_1 = 0.99$,$\varepsilon_2 = 0.999$ and $\varepsilon_3 = 0.9999$

Table 4.10 lists the optimization expected distortion (in PSNR, dB) in all cases. The results match very well with the results in exhaustive search.

BER	**Foreman** (dB)			**Football** (dB)			**Mobile** (dB)		
	1CL	2CLs	3CLs	1CL	2CLs	3CLs	1CL	2CLs	3CLs
1×10^{-3}	38.43	37.97	37.93	27.55	27.38	27.22	28.72	28.40	28.34
1×10^{-4}	38.52	38.40	38.05	27.66	27.52	27.28	28.84	28.70	28.40
1×10^{-5}	38.53	38.43	38.33	27.68	27.55	27.46	28.86	28.73	28.63

Table 4.10: The optimal expected distortion (PSNR, dB)of different sequences at different BER, where iCL(s) means encodes to i clusters.

The simulations in this section show that the optimization algorithm can converge both at single cluster coding and multiple cluster coding quickly.

4.4.2 FGA-FEC Wireless Extension vs. FGA-FEC

FGA-FEC wireless extension extends FGA-FEC by adding bit level protection to each description. We want to know if it is worth to extend the original scheme with bit level protection. We use the network topology shown in Fig. 4.14, where the encoded *Foreman* video sequence is sent from server to receiver over a lossy channel. The channel available bandwidth is set to 960 Kbps, packet drop probability is set to 0.05. We tested two BER scenarios: BER1= 6×10^{-5} and BER2= 1.1×10^{-4}. For wired FGA-FEC we only use RS codes as column codes. The packet-loss rate is calculated using (4.3), considering the channel BER. FGA-FEC is encoded based on available bandwidth $B = 960$ Kbps and the calculated packet-loss rate. For its wireless extension, we use product codes and the optimization algorithm proposed in this chapter.

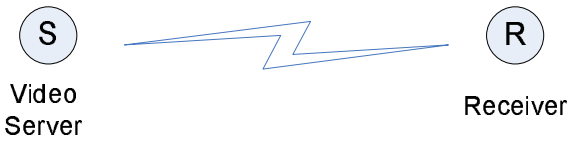
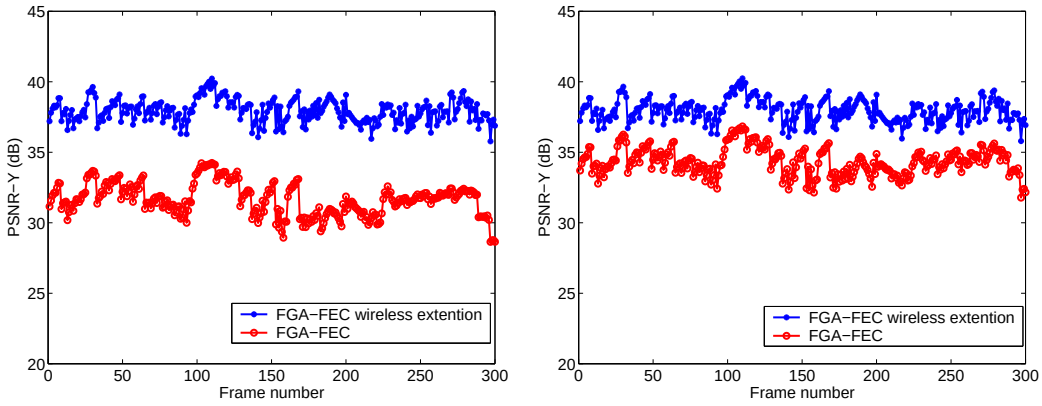


Figure 4.14: Topology of comparing FGA-FEC vs. its wireless extension. The encoded bitstream is sent from server to receiver through a wireless BER channel

Fig. 4.15(a) and 4.15(b) compare the PSNR of FGA-FEC vs. its wireless extension at different channels bit error rate. As shown, FGA-FEC wireless extension is much better than FGA-FEC over a wireless channel. At higher BER channel,

the wireless extension performs even better than FGA-FEC. This is because one bit error in a packet is treated as one erasure while decoding the column codes. Without bit level error correction codes, the packet erasure probability greatly increase in a wireless network, hence the optimizer needs to allocated more bandwidth to FEC codes and result in a lower delivered video quality. Therefore, it is necessary to extend the FGA-FEC to wireless network by adding product codes to correct bit errors.



(a) BER= 1.1×10^{-4} (b) BER=6×10^{-5}

Figure 4.15: Compare the PSNR of FGA-FEC vs. its wireless extension at different channels with different bit error rates.

4.4.3 Generalized FGA-FEC vs. Wireless MD-FEC in SNR Adaptation

We already show the efficiency of FGA-FEC adaptation in wired network in Chapter 3. Now we show how the generalized FGA-FEC adaptation works in a wireless network. We compare the generalized FGA-FEC vs. wireless MD-FEC to adapt to different bandwidth, by sending the encoded *Foreman* sequence to receiver with bandwidth ranging from 200 Kbps to 1000 Kbps as shown in Fig. 4.16, where node1 is the sender, node2 works as intermediate node which performs bitstream adaptation and BCH decoding/re-coding, node3 is the receiver. BER between node2 and node3 are set to 1×10^{-4}, $p_{\text{drop}} = 0.05$ at node2. There is no congestion between node 1 and node2.

For the generalized FGA-FEC, we first optimize the product codes and encode each GOP of *Foreman* to 64 descriptions with $p_{\text{drop}} = 0.1$, $p_b = 1 \times 10^{-4}$, $B = 1$

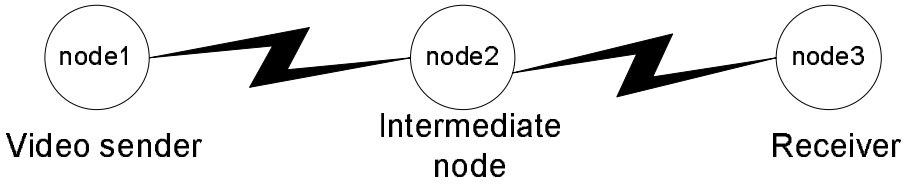


Figure 4.16: The topology of comparing FGA-FEC and MD-FEC over wireless lossy channel.

Mbps and then send the descriptions over the channel. For wireless MD-FEC, we apply the same amount of bit level protection as the generalized FGA-FEC to each encoded description. Fig. 4.17 shows the observed video quality (PSNR) at different available bandwidth. Clearly, FGA-FEC has much better performance in response to channel conditions. This is because MD-FEC responds to limited bandwidth between nodes2 and 3 by only dropping packets, hence some useless data is sent within remaining packets. On the other hand, FGA-FEC adaptation is performed actively by both packet shortening and packet dropping, and so avoid transmission of useless date, thus saving bandwidth for useful data and hence has better adaptation performance.

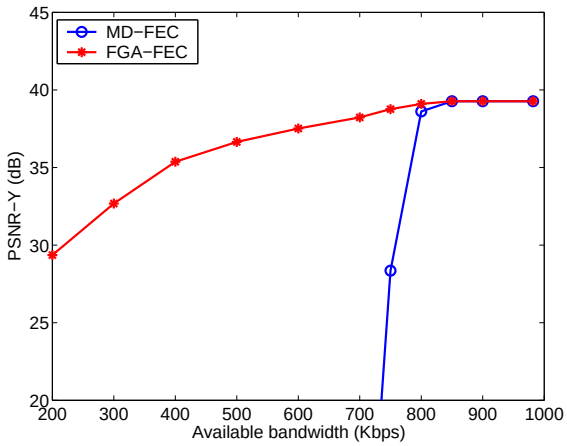


Figure 4.17: The generalized FGA-FEC vs. wireless MD-FEC at adaptation to different available bandwidth from 200 Kbps to 1Mbps over a lossy wireless channel.

4.4.4 Generalized FGA-FEC vs. Wireless MD-FEC in Frame-rate and Resolution Adaptation

In addition to SNR adaptation, the generalized FGA-FEC can do spatial and temporal adaptation as well. At Fig. 4.18, we set up a dynamic channel between node2 and node3 to test the adaptation capability of the generalized FGA-FEC and wireless MD-FEC, where the channel BER and bandwidth changes over time as shown in Fig.4.18.

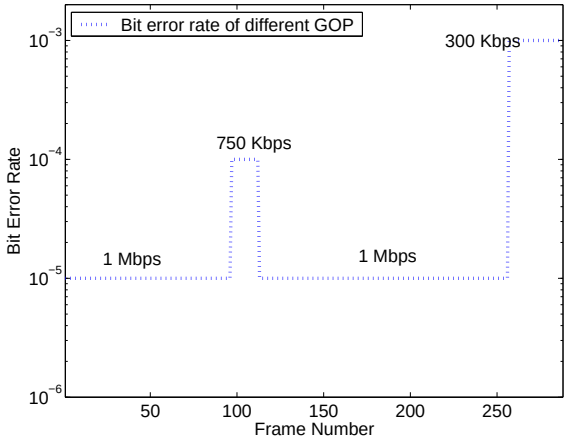
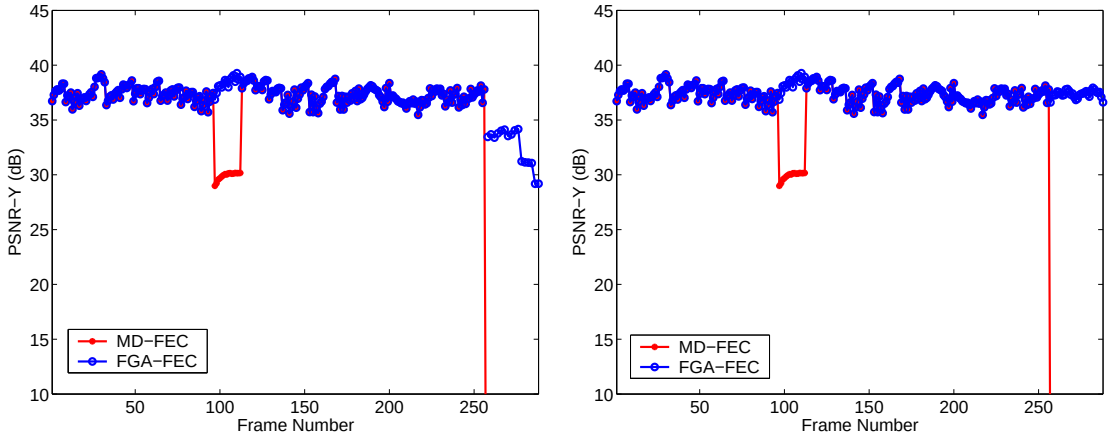


Figure 4.18: Channel conditions between node2 and node3

Again, we sent the Section 4.4.3 encoded sequence to the receiver. At the bad condition (1×10^{-4} /750 Kbps), both the generalized FGA-FEC and wireless MD-FEC use SNR adaptation. At the very bad channel state (1×10^{-3} /300 Kbps), FGA-FEC adaptation first does SNR adaptation, however, since this alone cannot satisfy the user requirement, our algorithm further does frame-rate adaptation by 2 (Fig. 4.19(a)) and/or resolution adaptation by 2×2 (Fig.4.19(b)), implemented by packet shortening at the fine-grained block level. Since wireless MD-FEC only drop packets in this very bad condition, even the video base layer can not go through the channel so that no video is decoded for the last two GOPs(frames 253-288).

The results in these two Sections show that the generalized FGA-FEC can provide near optimal protection but with much better adaptation performance.



(a) Frame rate (b) Resolution

Figure 4.19: Adaptation to different network conditions by frame rate and resolution.

4.5 Conclusion

In this chapter, we generalize FGA-FEC for embedded video bitstream protection and content adaptation over wireless channels and propose a fast search algorithm to assign the optimal product codes. Simulations show the efficiency for simultaneous content protection and adaptation.

CHAPTER 5
Improving Multimedia Throughput Using Header Error Protection in WLANs

In addition to congestion related packet losses, wireless networks have to deal with a time varying, error-prone, physical channel that in many instances is also severely bandwidth constrained. As such, protocol design, such as link-layer error-control may impact the performance of the network and the applications. In this study, we propose two link layer error protection schemes (header CRC and header FEC) that improve the effective throughout of wireless networks. Error control is applied to the packet header (at link level) and packet payload (at application level) separately. The network intermediate nodes either use header FEC or header CRC checksum to successfully transport the packets from the source to the destination. We compare the proposed schemes with conventional IEEE 802.11 protocol which is designed for reliable data communication. Both theoretical analysis and *ns-2* simulation results show that header error-protection strategy can effectively increase the application throughput.

5.1 Introduction

Throughput and bit-error rate are two important factors for wireless multimedia data transmissions. Many error control techniques have been proposed for both link and application layers. In this study, we focus on how to improve the application layer *effective throughput*, by providing header CRC and header FEC in the link layer. We define the effective throughput as the the fraction of channel bandwidth used by successfully transmitted packets.

Current IEEE 802.11 wireless LAN MAC protocol [87] is designed for reliable data transmission. One bit error in the link-layer packet could cause the drop of the whole packet at the receiver side, even though other bits of the packet are successfully received. We argue that this approach is not optimal for multimedia data delivery, since the bit errors could be corrected at application level by error-control codes or

the packet could be used directly by error resilient decoders. Therefore, in order to efficiently support multimedia data transmission, we propose a new wireless link-layer protocol, called HEP (header error protection), where we only protect the header part, and packets with errors are forwarded to the next node or passed up to the application layer. The main purpose of protecting the header is because it carries routing information for packet forwarding. Also the application header includes important information for multimedia bitstream which is critical when decoding. Therefore *header protected* in this paper treats both the IP and the application layer headers.

There are some interesting works on performance enhancement of multimedia transmission over wireless networks. For instance, in [88], layered video coding coupled with unequal error protection obtained by using different 'retry' limits at the link level has been shown to balance the link erasure rate and the overflow rate. Zheng and Boyce [109] propose a modified version of the UDP protocol to accommodate Internet-to-wireless video traffic. A new protocol stack design is proposed to allow bidirectional information exchange so that the physical, and link layers can communicate with the application layer. There are also arguments on whether error control should reside in the link layer or in the application layer [110]. Here, we provide another option - do part of the error control at the link level and leave some work to be done at the application level. Specifically we propose two header error protection schemes and analyze their impact on the throughput of the wireless networks.

The rest of this chapter is organized as follows: Header error protection strategies are introduced along with their throughput analysis in Section 5.2. In Section 5.3, we show our simulation results. The conclusion follows in Section 5.4.

5.2 Header Error Protection and Performance Evaluation

We consider three packet[2] error control schemes: IEEE 802.11, our proposed header CRC, and our header FEC scheme. The IEEE 802.11 MAC protocol uses

[2]In this study we talk about link-layer PDU, yet we still use the general term *packet* instead of *frame*.

a stop-and-wait ARQ with a positive ACK for each packet. The CRC checksum protects the whole packet, so we call this scheme *packet* CRC. Our proposed *header* CRC aims to protect just the header (including IP header and application header), not the whole packet. A packet retransmission is only triggered if an error detected in the header. In *header* FEC, we further apply forward error correction (FEC) to protect the header information from bit errors. Thus, the network might deliver the packet with an error in the payload. A retransmission is issued when FEC redundancy fails to correct any header errors.

Consider an *ad hoc* network with n nodes randomly located in a unit area domain. It was shown in [111] that under a protocol model for interference, the average hop number h can be given as $\sqrt{\frac{n}{\log n}}$, and each node in the network transmits at an average rate of $\frac{c}{\sqrt{n \log n}}$ bits/sec, where c is a positive constant. This result indicates a vanishing throughput performance as the network scales. The evaluation uses statistical approximation with these average values. All results are expected to hold with high probability. In this study, we use the binary symmetric channel (BSC) model with cross-over probability p and a binary Markov channel model. The BSC is a memoryless model where the noise bits are produced by a sequence of independent trials. Each has the constant probability $1-p$ of producing a correct bit and probability p of producing a bit error. p is then the bit error rate (BER) for the wireless link. Binary Markov channel is the first order binary Markov channel model (called Gilbert model [95], [112] for packet transmission). It was shown through analysis and simulation that a first-order Markov process is a good approximation for fading channels [113]. The model is described by the transition matrix,

$$\begin{bmatrix} 1 - p_{01} & p_{01} \\ p_{10} & 1 - p_{10} \end{bmatrix}$$

where p_{01} (p_{10}) is the probability that the transmission of the current bit is unsuccessful (successful), given that the previous transmission was successful (unsuccessful). It can be shown that $\frac{1}{p_{10}}$ represents the average length of a burst of errors, and the average BER is given by $\frac{p_{01}}{p_{01}+p_{10}}$.

5.2.1 Packet CRC in IEEE 802.11

In IEEE 802.11, usually there is a limit on the number of times a WLAN card can retransmit a packet (say 4 times). The single hop packet-error probability is defined as P_{e_1} for packet CRC. For simplicity, we use a simple random error channel in performance analysis, with bit error probability p. Since the errors are independent,

$$P_{e_1} = \sum_{i=1}^{N} (1-p)^{N-i} p^i \begin{pmatrix} N \\ i \end{pmatrix}, \tag{5.1}$$

where N is the packet length (in bits). First we assume there is no limit on the number of retransmissions. Given the probability of packet error P_{e_1}, the average number of retransmissions for a single hop has a geometric distribution with success probability $1 - P_{e_1}$. Thus the probability that the number of retransmissions (excluding the first transmission) in one hop is:

$$\mathbf{P}\{ret = i\} = P_{e_1}^i (1 - P_{e_1}), \tag{5.2}$$

If a flow only has one hop distance and the bandwidth is W bps, then the effective throughput of this flow is

$$F(h=1) = \sum_{i=1}^{\infty} \frac{W}{i} \mathbf{P}\{ret = i-1\} = \sum_{i=1}^{\infty} \frac{W}{i} P_{e_1}^{i-1} (1 - P_{e_1}) \tag{5.3}$$

Note $\frac{d}{da}(\sum_{i=1}^{\infty} \frac{a^i}{i}) = \sum_{i=1}^{\infty} a^{i-1} = \frac{1}{1-a}$ when $|a| < 1$, so $\sum_{i=1}^{\infty} \frac{a^i}{i} = \int \frac{1}{1-a} = c_0 - ln(1-a)$. Let $a = 0$ to solve the constant value we get $c_0 = 0$, then we have:

$$F(h=1) = -\frac{(1 - P_{e_1}) \ln(1 - P_{e_1})}{P_{e_1}} W \tag{5.4}$$

For multi-hop networks, the error statistics for each hop are independent. Now suppose a flow has experienced h hops, consuming bandwidth $W_1, W_2, ..., W_h$ on each

link, respectively. Then the aggregate throughput of this flow is

$$\begin{aligned} F(h) &= E[\frac{W_1}{i_1}] + E[\frac{W_2}{i_2}] + \cdots + E[\frac{W_n}{i_n}] \\ &= -\frac{(1-P_{e_1})\ln(1-P_{e_1})}{P_{e_1}}(W_1 + W_2 + \cdots + W_h) \end{aligned} \tag{5.5}$$

Since all the $W_1, W_2, ..., W_h$ add up to the network aggregate throughput $c\sqrt{\frac{n}{\log n}}$, summing up all the flows in the network gives the total aggregate throughput. Therefore, we have

$$A_1 = -c\sqrt{\frac{n}{\log n}}\frac{(1-P_{e_1})\ln(1-P_{e_1})}{P_{e_1}}, \tag{5.6}$$

where A_1 is the aggregated effective network throughput, given packet CRC method used in IEEE 802.11 link layer. Note in (5.5), we set the retry limit to ∞. In fact, the effect of a finite retry limit has diminishing return. For typical BER it is easy to prove that result of the 4 retry limit can be well approximated by (5.6). Therefore we use the results given by (5.6) for the standard IEEE 802.11 protocol and the proposed header CRC.

5.2.2 Header CRC

The probability that any error detected in header is

$$P_{e_2} = \sum_{i=1}^{k+r}(1-p)^{k+r-i}p^i \begin{pmatrix} k+r \\ i \end{pmatrix} \tag{5.7}$$

where k is the header size, and r is the CRC bits.

In a similar form with previously introduced packet CRC, the aggregate effective throughput of networks using header CRC is:

$$A_2 = -c\sqrt{\frac{n}{\log n}}\frac{(1-P_{e_2})\ln(1-P_{e_2})}{P_{e_2}} \tag{5.8}$$

Keeping in mind that the factor $-\frac{(1-P_{e_2})\ln(1-P_{e_2})}{P_{e_2}}$ is a monotone decreasing function of P_{e_2}. This factor decreases from 1 to 0 as P_{e_2} increases from 0 to 1.

This is consistent with heuristic expectations, because one expects the throughput to increase when packet-error probability decreases.

5.2.3 Header FEC

In this scheme, FEC redundant bits are added to protect header part of each packet. BCH codes are well known codes for binary data transmission, especially good for large block codes [114]. We add m protection bits to each header for error correction. For a t-error-correcting linear code, a BCH code is capable of correcting a total of 2^m error patterns, including those with t or fewer errors. So the probability that the decoder commits an erroneous decoding in one packet is upper bounded by

$$P_{e_3} \leq \sum_{i=t+1}^{k+m} (1-p)^{k+m-i} p^i \begin{pmatrix} k+m \\ i \end{pmatrix} \tag{5.9}$$

A packet is likely to fail to reach the destination unless it succeeds at each hop during the transmission . Given the probability that a packet will be dropped in one-hop transmission P_{e_3}, it is easy to get the aggregate throughput of the network using header FEC:

$$A_3 = c\sqrt{\frac{n}{\log n}}(1-P_{e_3})^h = c\sqrt{\frac{n}{\log n}}(1-P_{e_3})^{\sqrt{\frac{n}{\log n}}} \tag{5.10}$$

We append the FEC protection bits to the tail of the packet, since errors tend to be in burst. This way the header and the protection bits are less likely to be corrupt simultaneously. Numerical results show that protecting the header, 1 or 2 redundant bytes are enough in a channel with a moderate BER.

5.2.4 Comparison of the Effective Throughput of These Schemes

By comparing P_{e_1} with P_{e_2} and P_{e_3}, it is quite clear the latter two have much lower values, thus the proposed HEP schemes have the advantage in terms of effective throughput. Fig. 5.1 shows the throughput of the three schemes versus the number of nodes in a network. The parameters used in these plots are: $p = 5 \times 10^{-5}$, payload $N = 500 \times 8$ bits, header $k = 240$ bits, error correction bits $m = 8$, CRC

bits $r = 8$, and error correcting capability $t = 1$ bit. Some intermediary results are: $P_{e_1} = 0.1910$; $P_{e_2} = 1.5217 \times 10^{-4}$; $P_{e_3} = 7.5634 \times 10^{-5}$. The P_{e_3} used here is an upper bound number, i.e., the worst case scenario. The results are generated using MATLAB and based on the equation derived above. The factor c on the y-axis is the same as that in the above equations.

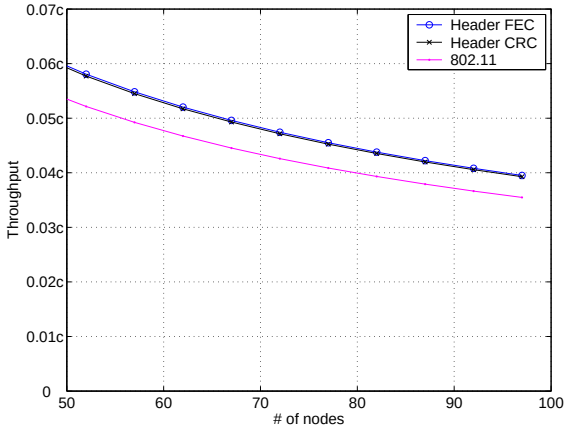


Figure 5.1: Per-node throughput as a function of n

Our analytical results are valid for large networks (large n). We choose the number of nodes from 50 to 100 in this plot to meet the practical scenario. Curves for header CRC and header FEC almost overlap over each other. The gap between the performance of these two HEP schemes and IEEE 802.11 indicates that the performance inefficiency of the current protocol can be improved using header error control. This result may help in the design of different protocol stacks according to different requirements. For applications having high requirements for data rate and lower requirements for accuracy of data, header error control is especially helpful. Header CRC is better for handling burst errors and header FEC can be adaptive to the link error environment (e.g., if the link error-rate increases, the protection bits can be added to correct more errors with little added cost).

The efficiency of coding requires the information message to be as small as possible. On the other hand, the more redundancy bits added, the more reliable the transmission would be. The question is how many bytes exactly we would encode. Considering that the IP header is 20 bytes, we now suppose 30 bytes are to be protected by error detection or correction, since there is important information

in headers from other layers as well. This header protection configuration can be adapted to different applications. For binary BCH codes, we choose codes that satisfy the block length of $k + m = 255$ bits, $k = 247$ bits, and $t = 1$ bit. This combination is the closest to 30 bytes (240 bits) header. We then use 8 error correction bits to correct single bit errors for 247 bits. So one byte extra can protect 30 bytes of header. Substituting these numbers in (5.9), we have $P_{e_3} = 2.9884 \times 10^{-5}$ and 3.0578×10^{-6} with $p = 10^{-4}$ and 10^{-5}, respectively. That means to protect the header no longer than 30 bytes, one byte is enough in this study. Also since P_{e_3} is so small, A_3 in (5.10) could be seen as asymptotically approaching $c\sqrt{\frac{n}{logn}}$ as $n \to \infty$.

5.3 Simulations

In this section, we evaluate our proposed HEP schemes using network simulator *ns2* [96]. We build new protocol models based on our proposed schemes and integrate them into *ns2*. The default packet-retry limits is set to 4 for long packets (such as a data packet) and 7 for short packets (such as control packets) - in both IEEE 802.11 and our proposed HEP protocols. We use RTP over UDP/IP with disabled checksum at a UDP segment (we call it an extreme case of UDPLite). The packet size used in all simulations is 500 bytes. In addition, all of our simulations use 2 Mbps radio.

5.3.1 Random Network Simulations

Following the set up in Section 5.2, we compare our proposed schemes with IEEE 802.11 in a random network. Nodes are placed randomly in a square domain, and the traffic pattern is random in this network. In *ns2*, the default setting of the antenna parameters results in an effective transmission range of 250 meters. The CBR rate of the random traffic is chosen in order to place the network in a saturation state. In this state, there is slight packet loss and if CBR rate is increased, the network aggregate throughput will not increase statistically. The routing protocol used is AODV [94].

We simulate random networks scaled from 50 nodes to 100 nodes under IEEE 802.11, header CRC, and header FEC protocol. The parameters in the Gilbert

model are $p_{01} = 2.5 \times 10^{-5}$ and $p_{10} = 0.5$, which yield an average channel BER 5×10^{-5}. The duration of each simulation is 2 minutes and the result is averaged upon 200 runs for different node distributions. The per-node throughput is shown in Fig. 5.2. The simulation results reflect statistically significant analysis based on a 95% confidence interval shown with error bars. If we compare Fig. 5.2 with Fig. 5.1, we see the plots share the same decreasing trend. The sharper decrease in the simulation results indicates the inefficiency of the MAC scheduling. When the network scales, the distributed MAC protocol can hardly give an optimal solution to achieve theoretical capacity. Nevertheless when the number of nodes is around 100, the average throughput improvement by using header CRC or header FEC over IEEE 802.11 is about 18%.

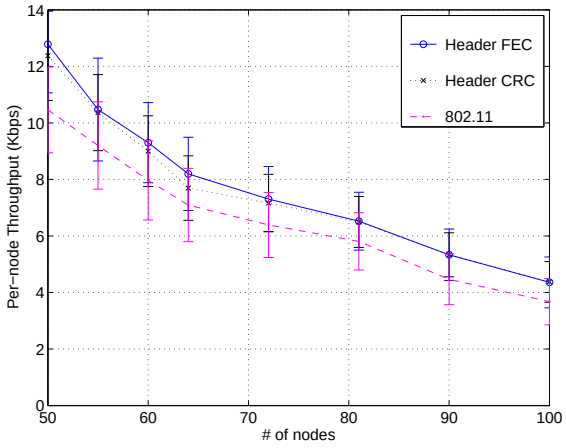


Figure 5.2: Simulation results on per-node throughput

5.3.2 Multi-hop Chain Scenario

Figure 5.3: A single chain with multi-hops from sender S to receiver R

The advantage of using HEP is more apparent in multi-hop networks, since retransmissions increase the traffic load and limit the throughput. In the following set of simulations we intend to find out the effective throughput of multi-hop networks under the three schemes. We use a single traffic chain model to avoid the effect of

the interference from other traffic. There are n nodes placed in a straight line, and neighboring nodes are separated by 200 meters, seen in Fig. 5.3. The multimedia traffic is sent from the first node towards the last node, traveling through all the intermediate nodes. Parameters of the Gilbert burst error model are: $p_{01} = 2.5 \times 10^{-5}$, and $p_{10} = 0.5$ at each hop. Fig. 5.4 illustrates our simulation results of the maximum throughput for a single chain. Let chain length be the number of nodes in a chain, each curve then represents the throughput of the different protocols when the chain length n increases from 5 to 10. The curve for IEEE 802.11 throughput performance is basically consistent with that in [115] (their throughput is a little bit higher since they do not have bit errors in their in *ns2* model).

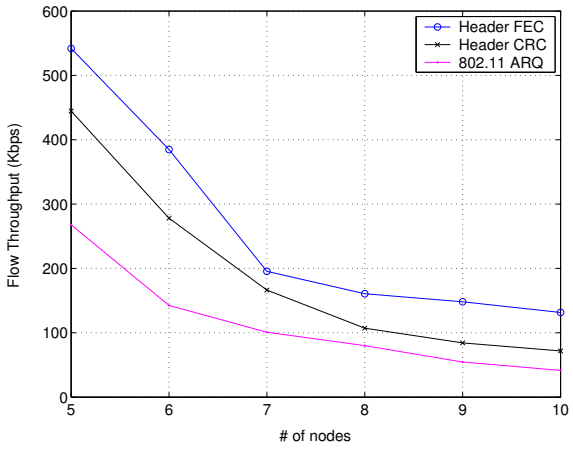


Figure 5.4: A single chain with multi-hops from sender S to receiver R

Header CRC performs not as good as header FEC, because headers cannot be recovered by FEC, there are still some packet drops due to retries exceed limit. Even though header CRC and header FEC consume some extra bandwidth for the application, the effective throughput is higher than that of 802.11. Simulation results show that there is some potential in throughput improvement for the header error protection schemes, especially when the network is large and hop number increases.

5.3.3 A Multi-hop Chain Topology with Cross Traffic

We place *ad hoc* nodes to model a linear topology with cross traffic. In this grid, shown in Fig. 5.5, the multimedia traffic is carried from node S to node R. Besides this main traffic, there is cross traffic from node S_1 to R_1, S_2 to R_2, ...,

and S_n to R_n, with n as the main chain length. This cross traffic also uses the main chain nodes as relays. The distance between each horizontally and vertically adjacent node is 200 m. Parameters of the Gilbert error model are $p_{01} = 1.25 \times 10^{-5}$, and $p_{10} = 0.5$.

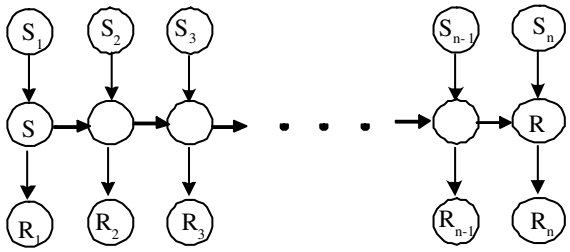
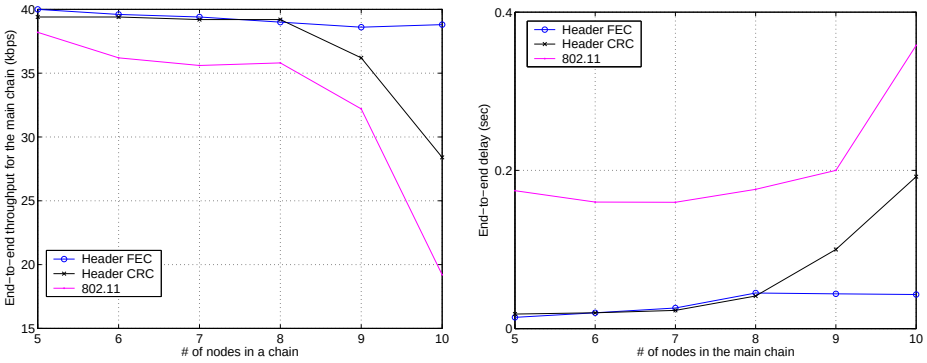


Figure 5.5: A chain topology with cross traffic

In this chain topology with cross traffic, nodes in the main chain have to compete with each other and with the nodes transmitting cross traffic. The RTS/CTS and DATA/ACK mechanism requires nodes near the transmitting node or receiving node to keep silent. If a transmitting node experiences a collision when sending RTS, it will choose an exponential random backoff time and retransmit after the backoff. The retransmission will continue if it fails, until it reaches the retry limit. It is obvious that the number of retries has impact on the end-to-end delay. It also increases the possibility of collision for the surrounding nodes, which further increases the delay of each transmission. The proposed header-protection strategy can largely reduce the number of retransmissions, thus it has the potential to reduce the end-to-end delay, which is important for a real time application. (Regarding the delay analysis, please refer to [116]).

Fig. 5.6(a) gives the result for the end-to-end throughput performance of the chain topology with cross traffic. The sending rate is at 40 Kbps, with the average channel error rate 2.5×10^{-5} at each hop. The cross traffic is also at 40 Kbps. The plot illustrates the flow throughput of the main chain (middle chain) traffic versus the chain length n.

The IEEE 802.11 MAC mechanism cannot discover the optimum schedule of transmissions on its own. Each node in a network experiences different degrees of competition. For example, nodes at the edges of the grid have fewer competitors



(a) End-to-end Throughput (b) End-to-end Latency

Figure 5.6: Performance of the chain topology with cross traffic

than those in the middle of the grid. So a portion of bandwidth is wasted by transmitting packets that are eventually dropped at middle nodes due to their higher degree of contention. Due to the limited retries and limited interface queue-buffer length, too many retransmissions, either from contention or packet CRC check, can make things much worse than the theoretical prediction would predict, especially near the network saturation point. That is why the original IEEE 802.11 scheme has a much lower performance than either header protection scheme.

5.4 Conclusions

This study proposes two HEP schemes, header CRC and header FEC, to improve the performance of multimedia transmissions. Both header CRC and FEC only need slight modification to the IEEE 802.11 MAC protocol. Network simulation results show that under a random network scenario, HEP takes advantage of FEC or ARQ to reduce the number of dropped packets at relaying nodes, thus can improve the throughput of the network. In this chapter, the link-layer protocol does not perform any error correction or detection for the packet payload. Therefore, we propose to use end-to-end error-control coding for the application layer, wherever it is needed. Application layer FEC is needed not only because of the channel errors, but also because of the packet losses caused by congestion. In the next chapter, we will investigate a cross-layer two-stage FEC scheme in conjunction with the proposed HEP scheme to streaming video over wireless networks.

CHAPTER 6
Two-Stage FEC Scheme for Scalable Video Transmission over Wireless Networks

In this study, we propose a two-stage FEC scheme with an enhanced MAC protocol especially for multimedia data transmission over wireless LANs. The proposed scheme enables the joint optimization of protection strategies across the protocol stack. In stage 1, packet-level FEC is added across packets at the application layer to correct packet losses due to congestion and route disruption. In stage 2, bit-level FEC is processed within both application packets and stage-one FEC packets to recover from bit errors in the MAC/PHY layer. Header CRC/FEC are used to enhance the MAC/PHY layer and to cooperate with the two stage FEC scheme. Thus, we add FEC only at the application layer, but can correct both application layer packet drops and MAC/PHY layer bit errors. We explore both the efficiency of bandwidth utilization and video performance using the scalable video coder MC-EZBC and ns-2 simulations. Simulation results show that the proposed scheme outperforms conventional IEEE 802.11.

6.1 Introduction

Current IEEE 802.11 wireless LANs are designed for reliable data transmission. They treat classical data flows and multimedia flows alike, even though these two kinds of flows have different requirements. The wireless physical (PHY) and media access control (MAC) [87] layers are designed to be as reliable as possible, so that one bit error in a packet could result in the whole packet being dropped. However, due to the error resilience features of many state-of-the-art multimedia CODECs and the utilization of error correction strategies at the application layer, packets with errors are still useful for multimedia applications. Therefore, mechanisms are needed to *efficiently* support multimedia data transmission over wireless networks. Packet losses in a wireless channel can be roughly categorized into two: (a) packets are dropped due to routing disruption, interference, and congestion in the intermediate

nodes, and (b) packets are discarded in the MAC/PHY layers due to internal bit errors. To efficiently protect data from losses/errors in a wireless environment, two questions occur: At which protocol layer should the protection scheme be located? and How should the protection strategies be deployed? One simple solution is to add protection mechanisms at each protocol layer, as in the current wireless 802.11 protocol. However, we argue that the layered protocol protection strategy does not always result in efficient performance for the delivery of multimedia data, due to the independency of each protocol layer.

In this study, we propose a two-stage FEC scheme with an enhanced MAC protocol to efficiently support multimedia data transmission over wireless LANs. Since only the application knows the characteristics of the multimedia data, the proposed scheme enables joint optimization of protection strategies across the protocol stack, and packets with errors are delivered to the application layer for correction or drop. The reason we choose to study FEC for video error recovery in this study is due to the fact that a wireless ad hoc network is usually multihop and multiple retransmissions would result in unpredictable delay and jitter at the application layer. We enhance the MAC/PHY layers to efficiently support multimedia flows by using both header CRC and FEC. We also slightly modified the protocol stack so that it can deliver packets with errors from the MAC layer to the application layer, instead of just dropping them. For the two-stage FEC, we add FEC only at the application layer, but can correct both application layer packet drops and MAC/PHY layer bit errors. Packet-level FEC (Stage 1) is added across packets at the application layer to correct packet losses due to congestion and route disruption. Bit-level FEC (Stage 2) is processed within both application packets and stage 1 FEC packets to recover bit errors from the MAC/PHY layers. Our proposed scheme has the following characteristics: *Network efficiency*: enhanced MAC protocol using header CRC and FEC improves application layer effective throughput; all useful information is delivered to the application. *Protection efficiency*: unequal error protection is easily deployable, since we only process FEC at the application layer. Furthermore, the proposed scheme combines bit-level protection codes (good at random bit error correction) and symbol level codes (powerful at correcting burst losses) to correct both

bit errors at MAC/PHY layers and packet losses at the application layer. Since we jointly consider the whole protocol stack, we can also call our proposed scheme cross-layer.

6.1.1 Related Work

In recent years, many papers have proposed cross-layer solutions for wireless video. Li and van der Schaar [88] proposed an error protection method that can provide adaptive quality of service to layered coded video by utilizing priority queueing at the network layer and retry-limit adaptation at the link layer. The video layers are unequally protected over the wireless link by the MAC with different retry limits that are dynamically adapted depending on the wireless channel conditions and traffic characteristics. Krishnamachari *et al* [89] propose an adaptive cross-layer protection strategy for enhancing robustness and efficiency of scalable video transmission where application layer FEC, MAC layer re-transmission strategy and an adaptive video packetization scheme are jointly optimized to maximize visual performance. The proposed scheme focus on wireless links from 802.11a base station to mobile users. Manshaei *et al* [90] propose a simple and efficient cross-layer mechanism for dynamically selecting the transmission mode that considers both the channel conditions and characteristics of the media. The proposed Media-Oriented Rate Selection Algorithm (MORSA) finds the highest possible transmission rate while guaranteeing a specific bit error rate by adjusting the physical layer modulation. Goldsmith *et al* [91] propose a cross-layer approach to support real-time video streaming, where information between different layers of the protocol stack is exchanged and end-to-end performance is optimized by adapting to this information at each protocol layer. Choi *et al* [92] proposed a cross-layer optimizer that interfaces the video streaming application and the radio link layer by means of parameter abstraction to maximize the end-to-end quality of the streaming service jointly for all users while efficiently using the wireless resources. State information is abstracted from selected layers and provided to the cross-layer optimizer.

6.1.2 Organization

The remainder of the chapter is organized as follows: In Section 6.2, we give a detailed description and analysis of our proposed two-stage FEC protection scheme and enhanced MAC protocol by using header CRC and FEC. In Section 6.3, simulation results are provided, followed by conclusions in Section 6.4.

6.2 System Overview

Fig. 6.1 illustrates the 802.11 wireless LAN protocol stack and packet structure associated with each layer, where "H"s represent the header of each protocol layer.

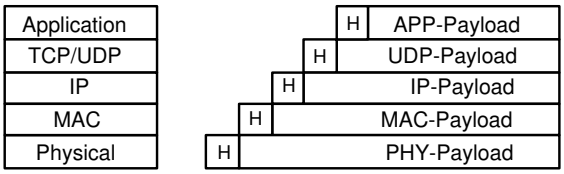


Figure 6.1: 802.11 protocol stack and associated packet structure

An application packet consists of data payload and application header. Whenever a packet is passed down to the next protocol layer, a header associated with that layer is added, as shown in Fig. 6.1. In this stack, UDP and IP provide source and destination IP addresses and port numbers of the communication pair to ensure correct delivery. Packets are dropped at the IP layer due to congestion or route disruption. On the other hand, MAC/PHY protocols support adjacent host communications and have to deal with bit errors. Any bit error within a packet could result in the whole packet being dropped, even though the errors could be corrected in the application layer. To efficiently support multimedia applications, we slightly modify the protocol stack so that it can deliver packets with errors to the application layer. This can be achieved by simply turning off the CRC checksum function in the MAC/PHY layers. The UDP-lite [93] protocol should be used at transport layer to match the enhanced MAC protocol. To ensure better delivery, we enhance the MAC/PHY layer by modifying the 802.11 packet CRC mechanism to check only the header part possibly also with bit-level FEC for the header part.

The proposed system diagram is shown at Fig. 6.2.

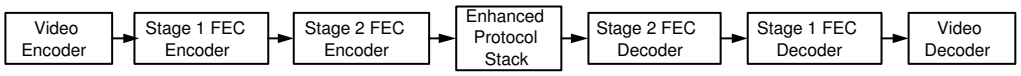


Figure 6.2: System diagram of the proposed two-stage protection scheme

At the application layer, two-stage FEC is applied to the encoded video bitstream based on network conditions. In stage 1, packet level FEC is added across application layer packets to correct packet drops due to congestion or route disruption. Stage 2 is processed within each application packet, a small amount of bit level FEC is added to recover bit errors from the MAC/PHY layers at each packet. At the receiver side, we first process the bit-level FEC, the bit errors from the MAC/PHY layers can be recovered. Then we pass the bitstream to the stage 1 FEC decoder for further correction. In this study, we choose Reed-Solomon (RS) codes for packet-level protection (stage 1) and BCH codes for bit-level protection (stage 2).

6.2.1 Channel Models and Enhanced MAC Layer

For simplicity, we start from a virtual channel with two nodes, one sender and one receiver. We further assume no contention between these two nodes. The binary symmetric channel (BSC) model and the Gilbert model [95] are used as our channel models. The Gilbert model is the first order binary Markov Channel model. Given two states, good state (G) with error probability P_G and bad state (B) with error probability P_B, the burst length in state G and B are both geometrically distributed with respective means P_{GB}^{-1} and P_{BG}^{-1}, where P_{GB} (P_{BG}) is the transition probability from the good (bad) state to the bad (good) state. The steady state probabilities of being in state G and B are $\pi_G = \frac{P_{BG}}{P_{GB}+P_{BG}}$ and $\pi_B = \frac{P_{GB}}{P_{GB}+P_{BG}}$. The overall average bit error rate p_b produced by the Gilbert model is:

$$p_b = P_G\pi_G + P_B\pi_B = \frac{P_G P_{BG} + P_B P_{GB}}{P_{GB} + P_{BG}} \tag{6.1}$$

The BSC error model is a memoryless model where bit errors are produced by a sequence of independent trials. Each bit has the probability p_b being flipped and $1 - p_b$ being successfully transmitted, p_b is then the Bit Error Rate (BER) for the

wireless link. Given a packet with size L bytes being transmitted over a wireless channel with BER p_b, the probability of packet error $P_e(L)$ can be calculated as:

$$P_e(L) = 1 - (1 - p_b)^{8L} \tag{6.2}$$

The 802.11 MAC layer defines two medium access coordination functions, basic *distributed coordination function* (DCF) and optional point coordination function (PCF). Since DCF can be used both in ad hoc and infrastructure modes while PCF can only work on infrastructure mode, we will focus on DCF mode in this study. DCF is a distributed medium access scheme based on the most popular *Carrier Sensing Multiple Access with Collision Avoidance* (CSMA/CA) protocol. The current MAC mechanism of 802.11 LAN uses stop-wait ARQ (SW-ARQ) to transmit a packet. If a packet arrives at a node with an empty queue and the medium has been found idle for an interval of longer than a distributed inter frame space (DIFS), the node can transmit the frame immediately, and the successful transmission of the packet is confirmed by an ACK packet. Therefore, both the packet itself and the feedback ACK must be successfully transmitted. Assume that the uplink and downlink have the same BER p_b, the probability to successfully transmit a packet P_{suc} is then:

$$P_{suc} = (1 - P_e(L))(1 - P_e(S_{ACK})) = (1 - p_b)^{8(L+S_{ACK})} \tag{6.3}$$

Where L and S_{ACK} are the size of MAC packet and ACK packet in byte, respectively. Given a physical layer bandwidth B_{PH}, the effective application layer throughput B_{AP} can be estimated as:

$$B_{AP} = B_{PH} * P_{suc} * r \tag{6.4}$$

where r is the ratio defined as $r = application\_packet\_size/MAC\_packet\_size$

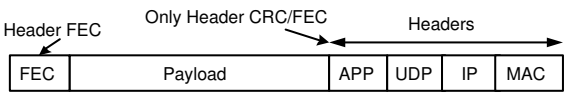


Figure 6.3: Enhanced MAC/PHY protocol using header CRC and header FEC

The header part of each protocol layer is crucial, because if header has some

errors in it, usually the whole packet is useless. We use header CRC and header FEC to enhance the MAC/PHY layers to efficiently support multimedia delivery. We slightly modified the 802.11 MAC/PHY layer packet CRC mechanism to check if there is something wrong within the header part as shown in Fig. 6.3. The packet is dropped if the header CRC fails. With this header CRC mechanism, the probability of successful transmission of a packet P_{sucH} becomes

$$P_{sucH} = (1 - p_b)^{8(S_{\text{header}} + S_{ACK})} \tag{6.5}$$

Where S_{header} is the size of all the header bytes. Since S_{header} is much smaller than the packet itself, the probability of successful transmission of a packet using header CRC is larger than when using whole-packet CRC. It also results in a larger effective throughput at the application layer according to Equation 6.4. Similarly to header CRC, a bit-level FEC can be added to the header part to combat bit errors in the header and further reduce the probability of header errors. We performed a MATLAB simulation to compare the application layer bandwidth efficiency of using header CRC, header FEC and packet CRC as shown at Fig 6.4. Here, we assume that the application layer packet payload size is 1000 bytes, the CRC header size is 60 bytes (UDP 8 bytes, IP 20 bytes, MAC header 24 bytes, application layer header 4 bytes) and the ACK packet size 14 bytes. Physical layer bandwidth is set to 2 Mbps. In all cases, a packet should be dropped if any CRC check fails.

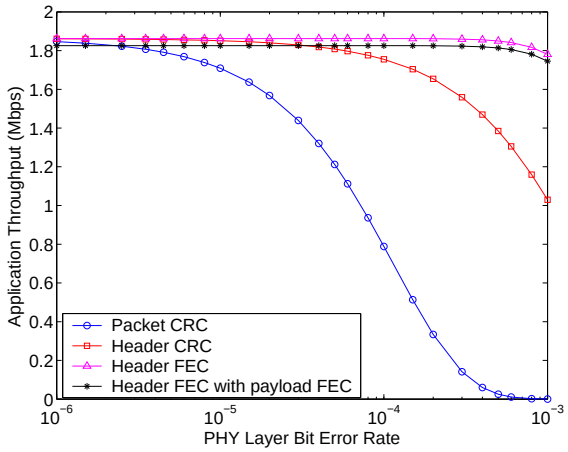


Figure 6.4: Application layer bandwidth efficiency vs BER

In Fig. 6.4, we use the same method as the current 802.11 does for MAC layer packet CRC, any bit error inside a packet results in the whole packet being dropped. For header CRC, only the header part (see Fig. 6.3) is checked at the receiver side, if anything is wrong within the header part, the whole packet is dropped. Even if only the header part is checked, the performance degrades a lot at high bit-error rates, and this is due to the large number of header check errors. So, we further added a $BCH(511, 502, 1)$ code to protect the header part from bit errors. The performance then becomes good even at high bit-error rates and the bandwidth overhead added by the header FEC is only 0.1%. Clearly, the header CRC/FEC results in a better application layer throughput, but the received packet may have errors in it. To protect the packet payload from errors, a $BCH(8191, 8000, 14)$ code is applied to each packet, and therefore, any 14 bit errors out of the 8191 codeword bits can be corrected. While fixed FEC adds overhead at low bitrates, it performs quite well at high bit error rates. In Fig. 6.4, a plot of header FEC with payload FEC has lower throughput compared with header FEC alone. This is because of the overhead in payload FEC and we also artificially drop the packet if payload FEC cannot correct the errors. This is comparable to 802.11 packet CRC with error free delivery. We will evaluate these schemes under more realistic conditions using the ns-2 simulator later on in this study. Here, we define a FEC decoding failure if FEC cannot correct all errors in a codeword. To identify a decoding failure is an engineering problem. If combined with CRC, the FEC decoder first decodes the codeword, then makes a CRC check of the decoded codeword, if the CRC is ok, then decode, otherwise, declare a decode failure.

6.2.2 Two-Stage FEC Scheme

At Fig. 6.4, even if a BCH code is applied to each packet payload, the curve with payload FEC still drops at high bit-error rate. This is because the whole packet is being dropped due to header FEC decode failure. Bit level in packet FEC protection cannot correct packet losses. Thus, we need to have a scheme to correct both bit errors and packet drops. We propose a two-stage FEC scheme to solve the problem as shown in Fig. 6.5.

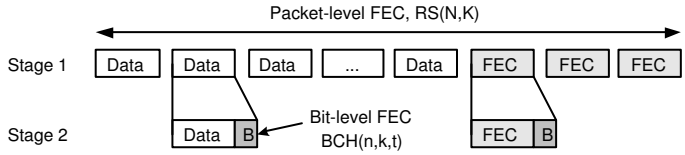


Figure 6.5: Detail of the proposed two-stage FEC scheme

In stage 1, packet level FEC is added across application layer packets to correct packet drops due to congestion or route disruption. We use RS codes for stage 1 FEC.

In stage 2, FEC is processed within each application packet, and a very small amount of bit-level FEC is added to recover any bit errors from the MAC/PHY layers. We use BCH codes for stage 2 FEC.

We assume a wireless channel with physical bandwidth B_{PH}, bit-error rate p_b, and probability of a packet being dropped at the sender due to congestion p_{drop}. For simplicity, we ignore ACK packet in this section. First we start from header CRC, since any bit error in header part would result in a whole packet being dropped, the probability of a packet loss p_{loss} can be calculated as:

$$p_{\text{loss}} = p_{\text{drop}} + 1 - (1 - p_b)^{8(S_{\text{header}})} \tag{6.6}$$

Bit-level FEC is added within each packet to correct bit errors. Given a BCH (n,k,t) code, number of bit errors larger than t in a codeword cannot be corrected, so the probability of not correctly decoding the codeword $P_{BCH}(E)$ is

$$P_{BCH}(E) = \sum_{j=t+1}^{n} \binom{n}{j} p_b^j (1 - p_b)^{n-j} \tag{6.7}$$

All these packets with errors are passed to the packet level FEC RS(N,K)for further correction. After BCH decoder correction, the residual bit-error rate p_{rb} can be estimated as:

$$p_{rb} = p_b P_{BCH}(E) \tag{6.8}$$

Reed-Solomon codes are Maximum Distance Separable (MDS) codes [79].

They are especially suitable for correcting burst errors. Given the correction results of BCH decoding, to calculate the probability of error on decoding the RS codeword, is a total probability problem. We define $R(E)$ as the event of RS decoder correction failure, B(E) as the event of BCH decoder correction failure and B(C) as the event of BCH decoder successful correction. Therefore the RS correction failure $P_{RS}(E)$ in the proposed two-stage FEC can be calculated as:

$$P_{RS}(E) = P\{R(E)|B(C)\}P_B(C) + P\{R(E)|B(E)\}P_B(E) \tag{6.9}$$

where $P_B(E)$ and $P_B(C)$ are the probability of BCH decoding error and the probability of BCH decoding success, respectively.

If BCH can successfully correct the bit errors inside packets, the conditional probability of RS error decoding is an erasure correction problem as

$$P\{R(E)|B(C)\} = \sum_{i=d_{min}}^{N} \begin{pmatrix} N \\ i \end{pmatrix} p_{syc}^{i}(1-p_{syc})^{N-i} \tag{6.10}$$

where the probability of symbol erasure is $p_{syc} = p_{\text{loss}}$, and $d_{min} = N - K + 1$.

If the BCH code fails to correct the bit errors inside the packets, then the conditional probability of RS error correction is a mixed erasure and error correction problem and can be calculated as

$$P\{R(E)|B(E)\} = \sum_{i=[(N-K)/2]+1}^{N} \begin{pmatrix} N \\ i \end{pmatrix} p_{sye}^{i}(1-p_{sye})^{N-i} \tag{6.11}$$

where the probability of symbol error is a combination of packet loss and packet error, can be calculated as

$$p_{sye} = p_{\text{loss}} + 1 - (1-p_b)^m \tag{6.12}$$

Where m is the symbol size of $RS(N, K)$ code.

After both BCH code correction and RS code correction, the residual bit error

Protection Method	FEC codes	Code rate
802.11	SW-ARQ	retransmission one time
RS only	RS(255,239)	239/255
Two-stage FEC with header CRC	BCH(8191,8000,14) + RS(255,245)	239/255
Two-stage FEC with header FEC	BCH(8191,8000,14) + RS(255,245) BCH(511,502,1)	239/255

Table 6.1: Parameter setups for compare of several protection schemes

rate can be reduced to

$$p_{\text{rsrb}} = p_{rb} P_{RS}(E) = p_b P_{BCH}(E) P_{RS}(E) \tag{6.13}$$

For header FEC, we can have a similar analysis, but using a residual bit-error rate after header FEC decoding to calculate p_{loss} at Equation 6.6.

We compare the protection performance of our proposed schemes (Two-stage FEC + header CRC/FEC) with conventional application layer FEC (RS only + 802.11) in terms of residual packet error rate. MAC layer re-transmission times are set to one at all three schemes. Any bit error in a packet after FEC correction should result in the packet being dropped, this is comparable to the situation in conventional 802.11 error-free delivery.

The parameter setup is given in Table 6.1. The packet size is the same as in Fig. 6.4. For RS only, we add FEC using RS code across packets and the code rate is 239/255. For 802.11, we do the same as in the 802.11 wireless LAN. Regarding two-stage FEC, we use $RS(255, 245)$ as stage 1 FEC and across the application layer packets. The $BCH(8191, 8000, 14)$ code is applied within each application layer packet as stage 2. Two-stage FEC with the header FEC scheme uses the same FEC for stage 1 and stage 2 as header CRC, but uses $BCH(511, 502, 1)$ as a protection method for the header part as shown in Fig. 6.3. The proposed two-stage FEC scheme significantly outperforms the conventional 802.11 plus application-only protection strategy as shown in Fig. 6.6

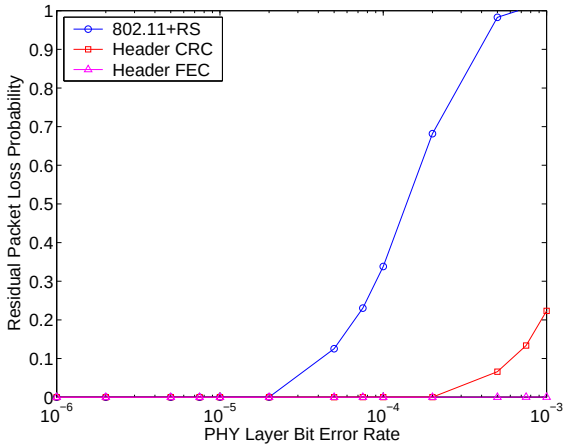


Figure 6.6: Residual packet loss probability of several FEC schemes vs BER

6.2.3 Effective Application-Layer Throughput

In this section, we analyze the effective application-layer throughput using different protection methods in a two-node communication topology, without contention. Here, we define the effective throughput as the throughput of error free traffic. Any packets with errors after correction are dropped at the application layer, and is comparable with 802.11 error -free delivery. We compare four protection schemes: 802.11 ARQ, application-layer FEC using Reed-Solomon codes, the proposed two-stage FEC with header CRC, and two-stage FEC with header FEC. We assume the same wireless channel as in Section 6.2.1.

For 802.11, any bit error in MAC packet should result in a whole packet being dropped. The effective application layer throughput $B_{AP}(802)$ can be estimated as

$$B_{AP}(802) = rB_{PH}(1-p_b)^{8(L+S_{\text{header}}+S_{ACK})}(1-p_{\text{loss}}) \tag{6.14}$$

In our application-layer scheme, an $RS(N,K)$ packet-level FEC scheme is applied across packets with code rate $C_{RS} = K/N$. After the RS code correction, the effective application-layer throughput can be estimated as:

$$B_{AP}(RS) = rB_{PH}(1-p_{rbrs})^{8L/C_{RS}}(1-p_{\text{loss}})C_{RS} \tag{6.15}$$

Where p_{rbrs} is the residual bit error rate after RS decoding and $p_{rbrs} = p_b P_{RSO}(E)$. The probability of error decoding the RS codeword $P_{RSO}(E)$ can be

calculated using equation 6.11 and equation 6.12.

Regarding the two-stage FEC with header CRC/FEC, we combine both the packet-loss correction capability and bit-level protection ability to maximize the overall system performance. We use $BCH(n, k, t)$ for bit-level protection within a packet and $RS(N, K)$ for packet-level protection. C_{our} represents the combined code rate using two-stage FEC scheme.

At receiver side, the BCH decoder first decodes the received packet. No matter whether the BCH decoder can fully correct the bit errors or not, it passes the packets to the RS decoder for further burst-loss correction and also packet-loss correction. The effective throughput can be estimated as

$$B_{AP}(our) = rB_{PH}(1 - p_{\text{rsrb}})^{8L/C_{our}}(1 - p_{\text{loss}})C_{our} \tag{6.16}$$

Equation 6.16 can be used for both header CRC and header FEC, but using different p_{loss}. The value p_{loss} can be calculated directly from equation 6.6 if header CRC is used. For header FEC, we still use equation 6.6 but replace of p_b with p_{rb} from equation 6.8.

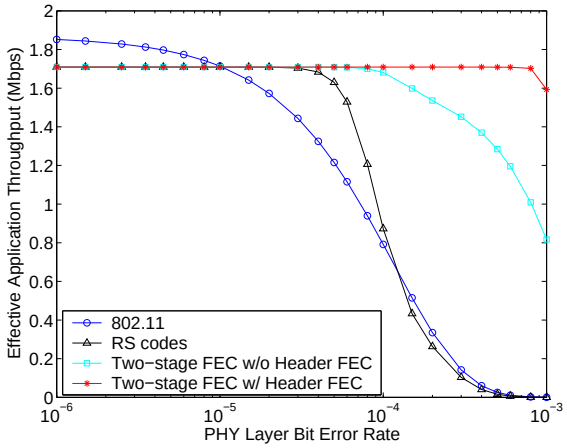


Figure 6.7: Effective application layer throughput efficiency of several FEC schemes vs physical channel BER

Fig. 6.7 shows the performance of the above mentioned four protection methods regarding their effective throughput. Except for 802.11, the same amount of FEC is added to data in the three other schemes. The simulation parameter setup

is listed in Table 6.1. The results show that at very low bit-error rate, 802.11 offers the highest effective throughput, since the SW-ARQ requires less overhead than fixed FEC protection. If an adaptive FEC scheme is used, we can expect similar results to those of our proposed scheme at low bit-error rates. As the bit-error rate goes higher, the performance varies dramatically. For 802.11, the probability of retransmission and packet drops goes very high at high bit-error rates, and it quickly reduces the effective throughput to a very low rate. Due to the characteristic of RS codes (if a codeword can correct the bit-errors, it would completely correct it or completely not correct it if errors are beyond its correction capability [79]), the RS-only protection method is even worse than 802.11 at high bit-error rates with the FEC overhead. On the other hand, our proposed two-stage FEC scheme effectively joins the advantages of both bit-level protection and packet-level protection. The performance is better than both 802.11 and RS-only protection schemes. The plot of two-stage FEC with header CRC also drops at higher loss rates, and this is due to the reason that at higher loss rates, the probability of header error goes up and results in a relatively high number of packets being dropped, beyond the correction capability of the RS code in the application layer. Further, we add a small amount of bit level FEC to protect header from errors. Since the added $BCH(511, 502, 1)$ can correct one bit error in a 511 bits codeword, the performance is very good compared to the other three schemes at high BER with less than 0.1% overhead.

6.2.4 Scalable Video Coding and FEC Design

The wireless channel is time varying, error prone, and usually bandwidth constrained. A distinct characteristic of wireless communications is its large variation in bandwidth and packet loss rate. Compared with the conventional fixed-bitrate video or multi-layer approach that only supports a discrete number of bitstream layers, scalable video coding is more suitable for wireless communications, since a scalable video bitstream can be almost continuously tailored to the time-varying channel characteristics. In this study, we use the fully scalable coder MC-EZBC [28] to evaluate our proposed two-stage FEC scheme, in conjunction with an enhanced MAC protocol.

6.2.4.1 MC-EZBC Coding

MC-EZBC is a highly scalable motion-compensated subband/wavelet video coder with high compression performance, rivaling that of the unscalable coding standard H.264. It produces embedded bitstreams supporting a full range of salabilities. Fig. 6.8 shows a typical Group-Of-Pictures (GOP) structure of this coder with 16 video frames. The top level represents the video at full frame rate. These incoming frames are subject to motion estimation and the resulting motion vectors (shown as arrows) are used for motion-compensated (MC) temporal filtering (MCTF). In this version of the coder, neighboring frames are decomposed using a motion-compensated Haar filter bank to produce the temporal low frequency bands (solid lines) and temporal high frequency bands (dashed lines) at the next lower level. This process is repeated until we obtain the MC average of all 16 frames in the GOP, which is at the bottom of the temporal pyramid. Video data in this case has five temporal scalability layers, going from full frame rate down to LLLL- level at 1/16 of full frame rate. Temporal subbands are then subject to spatial subband/wavelet analysis and encoded using a version of the EZBC coding algorithm, details of which are given in [28]. The bitstream sequence is organized in an embedded fashion. Each GOP coding unit consists of independent bitstreams $\{Q^{MV}, Q^{YUV}\}$, where Q^{MV} denotes the bitstream for the motion fields, and Q^{YUV} for the subband coefficients of color components Y, U, and V of the video. The motion vector code stream is embedded in frame rate. The remaining bitstream is fully embedded in quality/bitrate, spatial resolution, and frame rate. Such a scalable bitstream is especially suitable for mid-stream adaptation and can be adapted to different frame rates, SNRs, and resolution according users' requirements. For simplicity, we only consider SNR or bitrate scalability in this study. Scaling in term of quality is obtained by stopping the extraction process at any point in the bitstream. To achieve a certain bitrate, we simply stop extracting bits when that bitrate is reached.

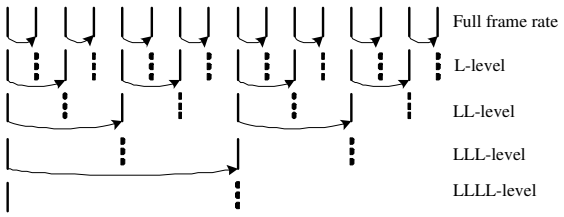


Figure 6.8: A typical GOP of 16 frames with 5 layers of temporal scalability

6.2.4.2 FEC Design

Since the wireless channel is time varying, the effective video bit rate correctly received at the receiver side is a random variable. The 802.11 wireless LAN MAC layer uses SW-ARQ to ensure packet delivery. Therefore, a sender can easily estimate its sending rate based on the ACKs. A video system is time sensitive, so excessively delayed packets are useless. The advantage of the scalable encoded bitstream is that it can be chopped at any point to match very well with the bandwidth varying channel: the more bits the receiver gets, the better the video quality. In this study, we use MD-FEC [52] as stage 1 FEC to protect the MC-EZBC video bitstream. Detailed description of MD-FEC can be found at Chapter 3. MD-FEC transforms this unequally important bitstream into one set of equally important descriptions (packets) by using erasure correcting RS codes. The benefit of using MD-FEC as stage 1 is that we can at least decode to a certain rate if any part of the bitstream is received. According to Equation 6.2, bit error rate at lower protocol level can dramatically affect the application layer throughput. Therefore the FEC design should try to recover all the random errors at the low protocol levels. Given the needed bit-level FEC bandwidth B_{bit} and total available bandwidth B_{avail}, the allocated bandwidth for MD-FEC, R_{max}, (please refer to Chapter 3, MD-FEC optimization algorithm) can be calculated as $R_{max} = B_{avail} - B_{bit}$. In this study, we use the method proposed in [55] to allocation optimal stage 1 FEC according to network conditions.

6.2.4.3 FEC Adaptation

To efficiently protect packets from losses and to match the available sending rate, adaptation is needed for FEC design. The FEC codes cannot only correct

errors, but also detect errors. The receiver estimates the loss behavior of the channel and feeds back the result to the sender. Two types of loss information are sent back to the sender. The packet loss information is fed back regarding stage 1 FEC design for each GOP. This loss information does not include packet drops due to FEC correction failure. Since bit errors in the packet can dramatically affect the application layer loss rate, stage 2 bit-level FEC uses a Step-Increase-Step-Decrease (SISD) method. A NACK packet is sent back to sender in the case of FEC decoder failure. Then the sender encodes the bit-level FEC with a step higher FEC code, eg. from $BCH(n, k, t)$ to $BCH(n, k, t+1)$. If errors inside a packet can be corrected, the receiver should also know how many bit errors are inside the packet. If the correction capability is much higher than the bit errors, for instance, the correction capability is twice higher than the number of errors, the receiver also feeds back an ACK for the bit-level FEC to step decrease one level from $BCH(n, k, t)$ to $BCH(n, k, t-1)$.

6.3 Simulations

To evaluate the performance of our proposed scheme in terms of effective application layer throughput and video PSNR, we perform several simulations to compare our two-stage FEC plus enhanced MAC protocol with the conventional 802.11 based method. The network simulator ns-2 [96] wireless module is used in this section and the simulation topology is shown at Fig. 6.9.

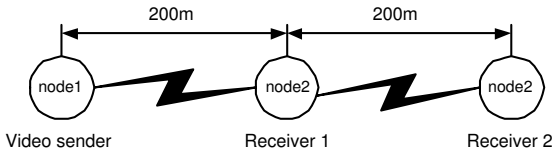


Figure 6.9: NS-2 video simulation topology

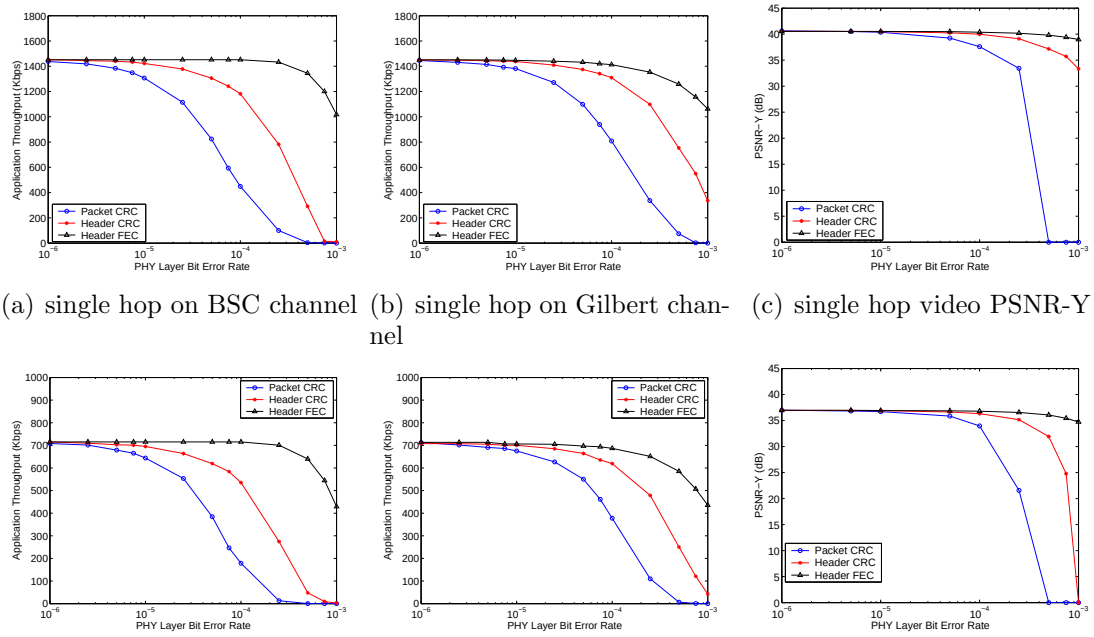
Two types of simulations are performed, single hop and multihop (2 hops in this study). In the single hop simulation, node1 works as sender, node2 as receiver, and node3 is idle. There is no contention in this scenario. For multihop simulation, node1 works as sender, node3 as receiver, and node2 is the intermediate node that forwards data from sender to receiver2. Contention exists among the three nodes. The wireless physical layer bandwidth is set to 2 Mbps. The bit-error rates in

this section are all average and the average bit-error burst length on the Gilbert channel is 2. In order to reduce delay variation, we set the maximum MAC layer retransmission time to 2. The retransmission is based on standard 802.11 SW-ARQ. Both RTS and CTS packets are used before a packet transmission. For clear illustration, we artificially set $PSNR = 0$ if there is no enough bandwidth to sent even the video base layer.

6.3.1 Effective Application Layer Throughput

To get the maximum effective throughput in the application layer, application layer CBR traffic is set to 2 Mbps from sender to receiver in single hop simulations, to saturate the channel. The packet and header size is set to the same size as in Section 6.2.1. To combat channel bit errors, a $BCH(8191, 8000, 14)$ code is applied to each packet in header CRC and header FEC. A packet is dropped upon BCH decoder failure. For the 802.11 packet CRC scheme, we directly follow the standard, a packet CRC is performed at receiver. Any bit error must result in the whole packet being dropped and trigger retransmissions until the maximum retransmission times. In the header CRC scheme, the receiver performs a header CRC, and drops a packet if the header CRC fails. In the header FEC scheme, a $BCH(510, 480, 3)$ code is applied to the 60 byte header part, resulting in 2 additional FEC bytes. This code can correct a number of bit errors up to 3 in a 511 bit codeword. If the BCH decoder cannot successfully decode the codeword, the a retransmission is triggered. In multihop simulations, since there are contentions among the three nodes, we reduce the application layer CBR traffic to 1.2 Mbps.

Fig. 6.10 shows the effective application-layer throughput on single hop simulation on the BSC channel (Fig. 6.10(a)), Gilbert channel (Fig. 6.10(b)) and multihop simulation on BSC channel (Fig. 6.10(d)), Gilbert channel (Fig. 6.10(e)). Similarly to Section 6.2.1, IEEE 802.11 performs very poorly at high bit error rates, because of the error-free-delivery design requirement. Compared to results in Section 6.2.1, the header CRC scheme performs worse than the theoretical simulation, this is because of the additional loss of RTS/CTS packets and ACK packets at higher bit error rates. With the help of header FEC, the probability of header error



(a) single hop on BSC channel (b) single hop on Gilbert channel (c) single hop video PSNR-Y

(d) multihop on BSC channel (e) multihop on Gilbert channel (f) multihop video PSNR-Y

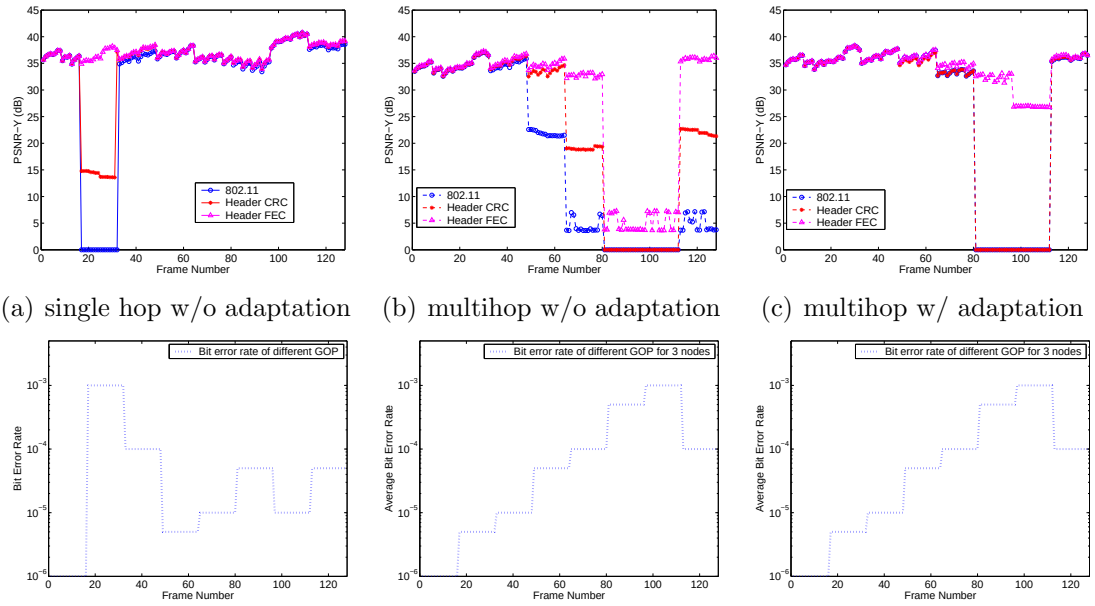
Figure 6.10: Effective application layer throughput on BSC and Gilbert channel at different physical layer BER and corresponding Video PSNR_Y

is greatly reduced. The degradation of the curve is most likely due to the ACK error and RTS/CTS failure at higher bit error rates. For example, at 1×10^{-3} bit error rate, the probability of ACK(14 bytes) error is around 10.6% and the RTS(20 bytes)/CTS(14 bytes) packets error probability is 23.8%.

Given the effective application layer throughput at Fig. 6.10(b), Fig. 6.10(e), we further test the performance of the video system. We assume an MC-EZBC encoded video bitstream is sent over a wireless Gilbert channel. The sender can adapt the bitstream based on channel conditions. The video sequence is monochrome *Foreman* CIF, 30 fps. The PSNRs shown in Fig. 6.10(c) and Fig.6.10(f) are the average of the first 100 frames from the single hop and multihop simulations. We notice that the PSNR for 802.11 packet CRC reduces to zero at higher loss rates, and this is thought due to there not being enough bandwidth for transmission of even the base layer of the bitstream. Clearly, we see better PSNR using our enhanced MAC protocol (header CRC and header FEC). The contention among the three nodes reduces the performance of the system.

6.3.2 Video Performance

We further tested the video performance of our proposed scheme using MD-FEC. Three kinds of simulations were performed: single hop simulation, multihop simulation without FEC adaptation, and multihop simulation with FEC adaptation. The MC-EZBC video bitstream was first encoded with MD-FEC at maximum bitrate 1 Mbps. Each GOP was encoded into 128 packets by the MD-FEC encoder for stage 1 FEC and resulted in a packet size of around 500 bytes. All packets are further encoded with bit-level FEC (stage 2), and a $BCH(4195, 4000, 4)$ code is applied in both single hop and multihop simulations. The physical layer average bit error rates for each GOP are set at Fig. 6.11(d), 6.11(e) and 6.11(f), under Gilbert channel. The corresponding PSNR of each GOP is shown above each BER graph in Fig. 6.11. The protection schemes compared are 802.11 packet CRC, header CRC, and header FEC, all with two-stage FEC.



(a) single hop w/o adaptation (b) multihop w/o adaptation (c) multihop w/ adaptation

(d) BER for each GOP, single hop (e) BER for each GOP, multihop w/o adaptation (f) BER for each GOP, multihop w/ adaptation

Figure 6.11: Video PSNR_Y vs. frame number at different channel conditions of each GOP

Since there is almost no contention in single hop simulation, the packet loss is most likely caused by bit errors in the wireless channel. We see dramatic performance drop in the 802.11 and header CRC schemes at severe bit error rate (1×10^{-3}) in

Fig. 6.11(a). This matches very well with the trend in Fig. 6.10(b), where 802.11 has less bandwidth even than required for the video base layer, and the header CRC scheme can only accept the video base layer. In multihop simulation without FEC adaption, node2 works as the intermediate node to forward packets to node3, both node1 and node2 are senders, and further node2 is also a receiver. In Fig. 6.11(b), the MD-FEC encoded video bitstream is fixed at 1 Mbps. The wireless channel is time varying and error prone, therefore, the stage 1 MD-FEC design is based on 10% packet loss rate and average error burst length is 2 packets, for better protection. Due to the limitation of physical bandwidth and high number of retransmissions at high bit-error rates, a large number of contentions and packet drops reduces the effective throughput greatly, and that results in a large video PSNR drop. Though MD-FEC is very powerful, as the channel BER goes high (1×10^{-3}), the probability of retransmission goes very high, and none of the three protection schemes work well. But still the proposed header FEC scheme can transmit part of the base layer at 1×10^{-3} BER. Fig. 6.11(b) also matches very well with Fig. 6.10(f). In Fig. 6.11(c) multihop simulation with FEC adaption, the FEC design is based on the feedback from the receiver and the actual sending rate. At high bit error rates, the sending rate goes down and FEC can be designed based on the available sending rate. The sender can truncate the scalable video bitstream to suite the channel. Therefore, comparing to Fig. 6.11(b), all curves in Fig. 6.11(c) have better performance in terms of video PSNR, especially two-stage FEC with header FEC, which performs very good even in the face of severe channel conditions (1×10^{-3}). Video clips related to Fig. 6.11 can be found at my website [97].

6.4 Conclusions

In this study, we propose a two-stage FEC scheme with an enhanced MAC protocol (header CRC/FEC) to efficiently support multimedia data transmission over wireless LANs. The proposed scheme enables the joint optimization of protection strategies across the the protocol stack. Two-stage FEC combines bit-level protection codes (good at random bit error correction) and symbol level codes (powerful at correcting burst losses) to correct both bit errors in the MAC/PHY layers and

packet losses in the application layer. Simulations show that the proposed scheme outperforms conventional IEEE 802.11. Future work will focus on joint source and network coding for video streaming over a mobile multihop network.

CHAPTER 7
Overlay Multi-hop FEC Scheme for Video Streaming

In this Chapter, we release the assumption of "no congestion between DSNs" in Chapter 3 and focus on the problem of providing lightweight support at selected intermediate overlay forwarding nodes to achieve increased error resilience on a single overlay path for video streaming. We propose a novel overlay multi-hop forward error correction (OM-FEC) scheme that provides FEC encoding/decoding capabilities at some intermediate nodes in the overlay path. Based on the network conditions, the end-to-end overlay path is partitioned into segments, and appropriate FEC codes are applied over those segments. Architecturally, this flexible design lies between the end-to-end and hop-by-hop paradigms, and we argue that it is well suited to peer-based overlay networks. We evaluate our work by both simulations and controlled Planet-Lab network experiments. These evaluations show that OM-FEC can outperform a pure end-to-end strategy by up to 10-15 dB in terms of video peak-signal-to-noise ratio (PSNR), and can be much more efficient than a heavyweight hop-by-hop strategy, in which all the overlay nodes along the path are involved in FEC computation.

7.1 Introduction

Most recently, peer-to-peer (P2P) architectures and overlay networks are gaining attention. Padmanabhan *et al.* [8] discussed the problem of distributing media content, both live and on demand, to a large number of receivers in a scalable way. They propose a solution called CoopNet for content distribution that combines aspects of infrastructure and P2P-based content distribution, wherein clients cooperate to distribute content, thereby alleviating the load on the server. CoopNet builds multiple distribution trees spanning the source and all the receivers, for its multiple description coded media content. Yeo *et al.* [65] proposed an application-level multicast overlay using peering technology and a lightweight gossip (active probing) mechanism to monitor prevailing network conditions and improve tree ro-

bustness. Clients can dynamically switch to other parents if they experience a poor QoS. In [66], Chu *et al.* explored the possibility of video conferencing using an overlay multicast architecture. Their constructed overlay spanning tree is optimized according to measurements of available bandwidth and latency among users, and can be modified by the addition of good links and the dropping of the poor links. The main goal of a resilient overlay network (RON) [67] is to enable a group of nodes to communicate with each other in the face of problems with the underlying Internet paths connecting them. RON detects such problems by aggressively probing and monitoring the paths connecting its nodes. If the underlying Internet path is the best one, that path is used and no other RON node is involved in the forwarding path. If the Internet path is not the best one, RON will forward the packet by way of other RON nodes.

Providing high-quality video streaming over the current best-effort Internet is a challenging problem due to video's characteristics such as high bitrate, delay variation sensitivity, and loss sensitivity. Streaming video and other media have been intensively studied in the past several years. From the channel coding perspective, forward error correction (FEC) schemes are considered to protect packets from channel losses, at the expense of increased bitrate. A Reed-Solomon (RS) code based unequal error protection scheme in conjunction with scalable video coding was proposed by Horn *et al.* in [68], where different FEC codes are applied to video base layer and enhancement layer according to channel conditions. Tan and Zakhor [69] proposed a layered FEC scheme as an error control mechanism in a layered multicast framework. By organizing FEC into multiple layers, receivers can obtain different levels of protection commensurate with their respective channel conditions. Distributed video streaming using multiple servers and FEC was proposed by Nguyen and Zakhor [70, 71] and Kim *et al* [72]. In [71], all packets are protected by fixed FEC codes and the proposed rate allocation algorithm adjusts the transmission rates of all senders in order to minimize the probability of lost packets. In [72], the optimal amount of redundancy is applied to each bitplane for sub-streams using a bitplane-wise unequal error protection algorithm. Performance characteristics of peer-based overlay networks are likely to be very different and highly variable

with respect to the traditional Internet or even managed overlay networks. However, their massive diversity, i.e. multiple overlay paths that can be harnessed, can compensate for the performance variability of any one path [73, 74]. In addition, lightweight support at intermediate nodes can improve single path performance. In this study, we focus on the latter problem and propose a novel overlay multi-hop FEC (OM-FEC) scheme for video streaming over peer-based overlay networks. The OM-FEC scheme dynamically partitions the end-to-end overlay path into segments according to its error characteristic, and provides appropriate error resilience over each segment. Here, we do not focus on overlay path construction and routing problems. Rather, we assume a peer-based overlay path has been pre-constructed and we focus on how to efficiently utilize it. We will henceforth use the term "overlay path" to denote the constructed path over a P2P network.

7.1.1 Scope and Assumptions

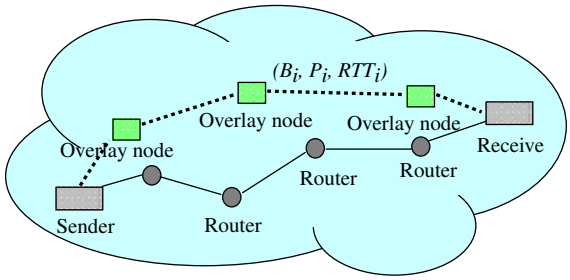


Figure 7.1: Streaming video using overlay network

Most prior work on video over P2P/overlay networks has focused on massive video data distribution or video conferencing using application-layer multicast. In contrast, our objective is to revisit the problem of efficiently utilizing the resources of a single overlay path. Our approach operates at small timescales in the data-plane, and can be combined with overlay routing and topology management approaches that operate in the control-plane and over larger time-scales [67]. In this sense, OM-FEC is complementary to prior work where resilience is provided using overlay routing methods. We assume that we can construct an overlay path with higher bandwidth [67] than the default Internet route by using P2P techniques such as Chord [75] or Pastry [49], to obtain a set of intermediate forwarding nodes as shown

in Fig. 7.1. In the figure, the dashed lines represent the virtual links between overlay nodes and the solid lines represent the default Internet path. In this study, we refer one virtual link as one "overlay hop." The quantities B_i, P_i, and RTT_i represent, respectively, the bandwidth, loss rate, and round trip time (RTT) of the ith hop.

7.1.2 Motivation

The advantage of application-layer overlay networks arises from two fundamental properties: (1) the overlay nodes have capabilities of computation and storage power that are far beyond basic store and forward operations, and (2) the overlay topology can be constructed and manipulated to suit one's purposes. Based on these two considerations, we argue that applying error correction purely end-to-end or hop-by-hop in an overlay network is a sub-optimal strategy. For example, in Table 7.1, we list a set of possible bandwidth and loss rates on a congested 6-hop overlay path based on the observations from [76, 77, 78], where B_i and P_i are, respectively, the bandwidth and loss rate on the ith hop. In [76], Boyce and Gaglianello found that the average packet-loss rate of sending a 1 Mbps MPEG video from Texas to Bell Labs (New Jersey) was 12.6%. Also, Tan and Zakhor [78] observed an average 18% packet-loss rate between Berkeley and Information Science Institute at Los Angeles. Here, we choose a moderate congestion case with end-to-end loss rate about 14%.

hop	1	2	3	4	5	6
B_i	660	625	615	700	900	1100
P_i	2.5%	3%	3.5%	2.5%	1.5%	1%

Table 7.1: An example of possible bandwidth (Kbps) and loss rate of an overlay path

FEC Method	end-to-end	OM-FEC	hop-by-hop
Throughput	529	594	594
Path loss rate	14%	14%	14%

Table 7.2: Path throughput (Kbps): OM-FEC vs. end-to-end and hop-by-hop FEC

Using FEC with RS erasure-correcting codes, in order to fully recover lost packets, the end-to-end based FEC scheme would have to be designed based on the end-to-end available bandwidth 615 Kbps and the end-to-end loss rate, approximately 14% in this case. Thus, the overall data throughput is reduced to around $(1-0.14)\times 615 = 529$ Kbps. On the other hand, if a heavyweight hop-by-hop based FEC scheme is used, the end-to-end data throughput will be 594 Kbps with the same path loss rate. However, the hop-by-hop FEC scheme induces more per-hop delay and uses more computational power of the overlay nodes than is necessary. Our proposed OM-FEC tries to minimize the overall computational complexity at the intermediate nodes, i.e. uses as few overlay nodes as possible to do FEC encoding/decoding, while still maintaining nearly the highest video quality that can be obtained over the overlay path. OM-FEC partitions the whole overlay path into segments and performs FEC over each segment.

This chapter is organized as follows. In Section 7.2 we describe our protocol, rate allocation scheme, and algorithms for the novel OM-FEC strategy. Then, we describe our simulation and controlled Planet-lab network experiments in Section 7.3. Finally, in Section 7.4 we make some conclusions and suggest possible extensions.

7.2 Overlay Multi-hop FEC (OM-FEC)

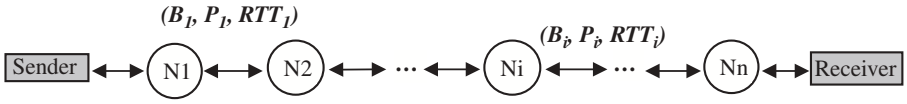


Figure 7.2: A sample overly path with n intermediate nodes

In our video streaming system, the video server sends a fixed rate bitstream to a user through an overlay path as shown in Fig. 7.2. There are n intermediate overlay nodes (each denoted as N_i), where hop i has available bandwidth B_i, packet loss rate P_i, and round trip time RTT_i. In order to protect packets from channel loss, FEC codes are deployed. Obviously, applying FEC over each hop results in the best video quality at the receiver, but also the most intermediate FEC encoding/decoding computation. On the other hand, an end-to-end based FEC scheme results in the

worst video quality but the least FEC computation. Our objective is to efficiently utilize the resources of the overlay path and reduce the overall FEC computational complexity while still maintaining near-highest possible video quality.

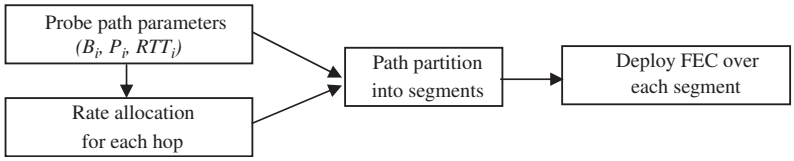


Figure 7.3: OM-FEC building blocks and the relationship among these blocks.

The basic building blocks of the OM-FEC scheme include an algorithm to determine optimal partitioning of the overlay path, a rate allocation algorithm for allocating appropriate FEC rate for different hops of the overlay path, and the actual deployment of the FEC on the path as shown in Fig. 3. The video server actively sends out a probe packet every Δt seconds. Each overlay node measures the loss rate and RTT of its related hop using this small probe packet. The obtained per-hop RTT and loss rate estimates are used to infer the TCP-friendly [6] available bandwidth of each hop. With this available bandwidth and loss rate, the FEC rate for each hop can then be calculated. However, different FEC coding is not needed for every hop. So, our server runs a greedy algorithm to partition the overlay path consistent with the above FEC rate estimates, so that the overall computational complexity at intermediate hops is minimized without sacrificing FEC-based resilience gains. This partitioning splits the overlay path into segments, and separate FEC encoding/decoding is then employed over the segments. Hence, only the boundary nodes between segments need to do FEC encoding/decoding. When this path partition algorithm produces a single segment (equivalent to the entire end-to-end overlay path), then FEC is designed based on the end-to-end network characteristics, and the overlay nodes simply receive and forward data and FEC packets onto the destination. The decision made by the server is conveyed to every node by a small command packet sent out from the server, so each node knows what it should do after it receives a command packet. The following sections outline the details of our OM-FEC scheme step-by-step.

7.2.1 Probe Network Parameters

Given the overlay path shown in Fig. 7.2, the server first needs to know the available resources of the path and then must decide how to efficiently use these resources. In OM-FEC, as mentioned above, an active probing method is used to estimate the RTT and loss rate of each hop. In order to synchronize overlay parameter calculation and reduce overhead bandwidth, the sender uses a small active probing packet to synchronize the estimation procedure. The probe packet is sent from the server every time interval Δt seconds. Each overlay node processes the probe packet, and calculates the loss rate and the RTT.

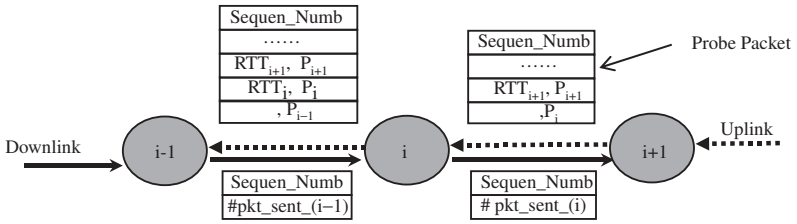


Figure 7.4: Probe packet information and processing; server sends out a probe packet downlink, the collected information of each hop is conveyed uplink to the server.

The structure of the probe packet is shown in Fig. 7.4 where the circles denote overlay nodes, $Sequen\_Numb$ is the sequence number of the probe packet, $\#pkt\_sent(i)$ denotes the number of packets in the ith node sent at the current calculation period. The parameters $\{RTT_i, P_i\}$ denote the RTT and loss rate of ith hop, respectively. This probe packet passes through all the nodes of the constructed overlay path. For *downlink* (link from server to receiver) path-parameter estimation, node i caches the probe packet from its *uplink neighbor*, replaces the item $\#pkt\_sent(i-1)$ with its own $\#pkt\_sent(i)$, and then forwards the probe packet to the next uplink node $(i+1)$. Each node records the time T_{send} – the instant when it sends the probe packet to its *down node*, and this parameter is later used for RTT calculation. With the received item $\#pkt\_sent(i-1)$ and the measured received data packets $\#pkt\_recvd\_from(i-1)$, the ith node can calculate the loss rate of the $(i-1)$th link as $P_{i-1} = (\#pkt\_sent(i-1) - \#pkt\_recvd\_from(i-1))/\#pkt\_sent(i-1)$ The probe packet is fed-back to the server after it reaches the receiver. This feed-

back packet collects all information from these overlay nodes while going back to the server along the uplink (link from receiver to server). As for RTT estimation, as soon as a probe packet arrives at the ith node from the $(i+1)$th node, the ith node obtains the arrival time of this packet T_{arrive}, and then calculates the RTT for the ith hop as follows:

$$RTT_i = T_{arrive} - T_{send} - \sum_{j=i+1}^{j=L} RTT_j \tag{7.1}$$

where L denotes the total number of hops between server and receiver. The ith node attaches its calculated loss rate for the $(i+1)$th hop and the RTT of the ith hop to the probe packet and then forwards it uplink to the $(i-1)$th node as shown in Fig. 7.4, until it eventually arrives at the server. Thus, the server has the loss rate P_i and RTT_i of each hop along the overlay path. The probing packet consumes extra network bandwidth. In order to save network resources, the probing packet should be small. Since it is very unlikely that one hop has a loss rate greater than 25.5% and RTT longer than 255 ms, we use two bytes to represent the $\{RTT_i, P_i\}$ pair, four bytes to represent $Sequen\_Numb$ (two bytes) and $\#pkt\_sent(i)$ (two bytes). Therefore, the bandwidth consumed downlink is $(4 \times 8)/\Delta t = 32/\Delta t$ (bps). The probe packet collects path information from each overlay node while traveling back from the receiver to the server. At each node, two bytes are appended to the probe packet. Thus the uplink bandwidth consumed at the hop adjacent to the receiver is $(1 \times 16 + 8)/\Delta t = 24/\Delta t$ (bps) and the largest uplink bandwidth consumed at the hop adjacent to the server is $(L \times 16 + 8)/\Delta t$ (bps). Given $\Delta t = 0.2$s, $L = 10$, the total probe downlink bandwidth is 0.16Kbps and the uplink bandwidth is in the range of [0.12, 0.84] Kbps.

7.2.2 Rate Allocation Strategy of OM-FEC

The video server has a fixed rate bitstream of rate B_{fixed}, to be sent to a user through an overlay path. At a certain time t, the server estimates the available bandwidth $B_{\text{avail}}(i,t)$, of each hop by using the information brought back in the probe packet. Then, OM-FEC uses a very simple method to allocate the available

bandwidth for both FEC and video data on each hop: for the ith hop, the algorithm assigns a portion of the available bandwidth $B_{\text{avail}}(i,t)$, to video data $B_{\text{data}}(i,t)$; and the remaining bandwidth is assigned to FEC $B_{FEC}(i,t)$, until either the desired FEC rate $B_{req}(i,t)$, is met or the available bandwidth budget is exhausted. In an extreme case, if $B_{\text{avail}}(i,t) \leq B_{\text{fixed}}$, all the available bandwidth is assigned to the video data, i.e $B_{\text{data}}(i,t) = B_{\text{avail}}(i,t)$. The main goal of the rate allocation scheme is to find $B_{\text{data}}(t)$ for the whole path and $B_{FEC}(i,t)$ for each hop based on its measured loss rate and RTT, for further utilization by OM-FEC (to decide what kind of FEC scheme should be deployed, OM-FEC, end-to-end FEC, or hop-by-hop FEC).

The bitrate $B_{\text{avail}}(i,t)$ (in bps) or TCP-friendly bandwidth of the ith hop is calculated using [6]:

$$B_{\text{avail}}(i,t) = \frac{S}{RTT_i\sqrt{\frac{2p(i,t)}{3}} + T_{rto-i}(3\sqrt{\frac{3p(i,t)}{8}})p(i,t)(1+32p(i,t)^2)} \tag{7.2}$$

where S is the packet size in bits, RTT_i is the estimated RTT of the ith hop in seconds, T_{rto-i} is the TCP timeout on the ith link, and $p(i,t)$ is the estimated loss probability for the ith link.

After the available bandwidths $B_{\text{avail}}(i,t)$, of all the hops are calculated, the end-to-end bandwidth $B_{e2e}(t)$, from source to receiver is determined as the minimum of the per-hop TCP-friendly bandwidths of the overlay paths, i.e.

$$B_{e2e}(t) = \min_{1 \leq i \leq L} \{B_{\text{avail}}(i,t)\} \tag{7.3}$$

Thus, the server decides the bandwidth allocated to video data for each hop as

$$B_{\text{data}}(t) = \min(B_{e2e}(t), B_{\text{fixed}}) \tag{7.4}$$

We use systematic Reed-Solomon erasure codes $RS(n_i, k_i)$ to protect packets from channel losses on hop i. Given the target loss probability P_{target}, and channel packet loss rate $p(i,t)$, if k_i is fixed, the lower bound of n_i can be determined from (7.5) [79]. According to the results presented in [80], the viewing quality of MPEG-4

encoded video is acceptable at a loss rate of 10^{-5}, and good at a loss rate of 10^{-6}. In this study, we choose the target loss probability $P_{target} \leq 10^{-7}$ at each hop

$$P_{target} = \sum_{j=n_i-k_i+1}^{n_i} \left(\begin{array}{c} n_i \\ j \end{array} \right) p(i,t)^j (1-p(i,t))^{n_i-j} \tag{7.5}$$

Due to the randomness of the network, the FEC calculated based on the network condition at time slot t may be not enough to protect the data from losses at time slot $t+1$; Therefore, we add an extra safety factor ε to the calculated FEC. ε can be chosen based on distribution of packet loss probability and the degree of fluctuation of the channel. In this study, we use experimental results to choose its value for a given channel. Thus, given the allocated video data bandwidth $B_{\text{data}}(t)$, and the estimated FEC, $n_i - k_i + \varepsilon$, on each hop, the required minimum FEC bandwidth $B_{req}(i,t)$ to recover the lost packets of each hop becomes:

$$B_{req}(i,t) = \frac{n_i + \varepsilon}{k_i} B_{\text{data}}(t) - B_{\text{data}}(t) = \frac{n_i - k_i + \varepsilon}{k_i} B_{\text{data}}(t) \tag{7.6}$$

Based on our statement at the beginning of this section, we can obtain the allocated bandwidth for FEC on each hop:

$$B_{FEC}(i,t) = \min\{B_{req}(i,t), (B_{\text{avail}}(i,t) - B_{\text{data}}(t))\} \tag{7.7}$$

$$B_{FEC}(i,t) \leq B_{req}(i,t) \quad \forall i = 1, 2, \cdots, L \tag{7.8}$$

If inequality (7.8) holds for all the hops, that means none of the allocated FEC bandwidth of these hops can fully recover the lost packets on its link. Then FEC should be added hop-by-hop, and in this case, OM-FEC works the same as hop-by-hop FEC, i.e. every intermediate node performs an FEC encoding/decoding computation. Given the available loss rate on each hop $p(i,t)$, the end-to-end loss rate $p_{e2e}(t)$ can be estimated using:

$$p_{e2e}(t) = 1 - \prod_{i=0}^{i=L} (1 - p(i,t)) \tag{7.9}$$

Similar to (7.6), the required end-to-end FEC bandwidth $B_{e2ereq}(t)$ to achieve the end-to-end loss rate can be calculated as

$$B_{e2ereq}(t) = \frac{n-k+\zeta}{k} B_{\text{data}}(t) \tag{7.10}$$

where n, k are calculated using (7.5) with an end-to-end loss rate $p_{e2e}(t)$, and ζ is an added extra safety factor for end-to-end FEC. The value of ζ can be chosen using the same method as ε.

$$B_{e2ereq}(t) \leq B_{e2e}(t) - B_{\text{data}}(t) \tag{7.11}$$

If inequality (7.11) holds, it means the available end-to-end bandwidth $B_{e2e}(t)$ is large enough for both video data $B_{\text{data}}(t)$, and the end-to-end required FEC bandwidth $B_{e2ereq}(t)$, then OM-FEC just adds FEC at the path ends, i.e. works the same as an end-to-end FEC scheme. No intermediate overlay node needs FEC encoding/decoding.

In intermediate cases between the end-to-end and hop-by-hop extremes, OM-FEC must partition the overlay path.

7.2.3 Overlay Multi-hop FEC (OM-FEC)

We want to minimize the overall FEC computational complexity at the intermediate nodes, while still maintaining near-highest video quality, i.e. the expected video distortion of using OM-FEC $E[D_{OM-FEC}]$, should be the same as that obtained by using hop-by-hop FEC $E[D_{hop-by-hop}]$. In order to maintain high video quality, we use a rate allocation algorithm to be described in previous Section 7.2.2. To minimize the computational complexity of the overlay nodes, OM-FEC should use as few nodes as possible for the FEC encoding/decoding. The OM-FEC partition algorithm should find the minimum number of segments N_{segment}, of the overlay path. Mathematically, the problem can be stated as follows:

$$\text{Minimize}(N_{\text{segment}}) \tag{7.12}$$

subject to:

$$\begin{cases} E[D_{OM-FEC}] = E[D_{hop-by-hop}]; \\ B_{\text{data}}(t) = \min(B_{e2e}(t), B_{\text{fixed}}); \end{cases}$$

Given a certain bitstream with bitrate B_{fixed}, OM-FEC attempts to find the best partition of the overlay path, i.e. the minimal number of partition segments N_{segment} that can maintain the same video quality as hop-by-hop. We use a forward partition approach to find a good partition of the path.

In order to reduce the computational burden, OM-FEC partitions the overlay path into segments according to the characteristics of each hop as shown in Fig. 7.5. For example, the OM-FEC algorithm partitions the overlay path into N segments, with J nodes in segment 1, L nodes in segment 2, up till M nodes in segment N. FEC is then deployed over each segment. The parameters J through M are dynamically determined by the OM-FEC algorithm as explained below.

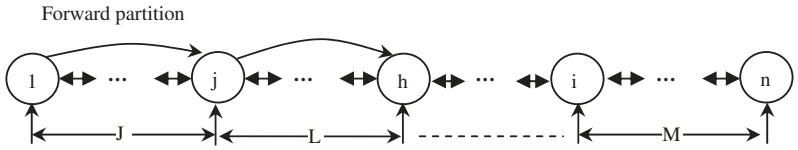


Figure 7.5: An overlay path is partitioned into segments by OM-FEC to reduce computational complexity at intermediate nodes. Only boundary nodes perform FEC encoding/decoding. Circles denote overlay nodes.

Since OM-FEC does a forward search to find the best partition of the overlay path, the server is the start node to partition the overlay path. The forward partition is defined as partitioning the overlay path along the direction from server to receiver. In Table 7.3, we define the terms which will be used in the partition algorithm. The computation unit for this part is based on a segment. For instance, $B_{FEC(start,i)}$ is calculated based on the method of Section 7.2.2 and (7.2)-(7.7), but considering the segment from the start node to the $(start + i)$th node as one virtual link and using the accumulated loss rate and available bandwidth of this virtual link.

The Forward Partition algorithm is described by the following pseudo-code given the server as the starting node. The output of Algorithm 6 is the partitioned segments of the overlay path and the amount of FEC that should be deployed over these segments.

Terms	Meaning	Calculation Equations
$B_{FEC(start,i)}$	Allocated FEC bandwidth for the segment from the start node to $(start+i)$th node	(7.2)-(7.7)
$B_{req(start,i)}$	Required FEC bandwidth for the segment from the start node to $(start+i)$th node	(7.9)-(7.10)
N_{segment}	Number of segments partitioned	

Table 7.3: Terms used in partition algorithm

Algorithm 6: Forward partition algorithm

Input : $B_{\text{avail}}(i,t)$, $B_{\text{data}}(t)$, P_{target}, n, $B_{req}(i,t)$

Output: Partitioned segments, B_{FEC} for each segment

//Begin calculation from the server;

$Start = 0$; $N_{\text{segment}} = 1$;

for $(i = 1; i \leq n; i++)$ **do**

  // Calculate the FEC bandwidth from start node to $(start+i)$th node;

  Calculate $B_{FEC(start,i)}$;

  //Calculate FEC bandwidth from start node to $(start+i+1)$th node;

  Calculate $B_{FEC(start,i+1)}$;

  //Find a boundary node to partition the path into segments;

  if $((B_{FEC(start,i)} \geq B_{req(start,i)})$ && $(B_{FEC(start,i)} \geq B_{req(start,i)}))$ **then**

    Start node to $(start+i)$th node is partitioned as one segment;

    $N_{\text{segment}}++$;

    FEC is deployed over this segment, with bandwidth $B_{FEC(start,i)}$;

    Start from boundary node to partition the rest of the path;

    $Start = Start + i$;

  end

end

The server runs this algorithm to partition the overlay path into segments and deploys FEC over each segment. The decision is then conveyed to all intermediate nodes by a small command packet. For each boundary node, the command packet contains a three-byte field specifying the node ID, and the n and k parameters of the chosen RS (n,k) code. The nodes whose IDs are not listed in the command packet will simply forward all the packets they receive, without FEC coding/decoding.

Thus, each node knows what it should do after it receives the command packet. Based on the OM-FEC strategy, the largest segment could include all the nodes of the overlay path (same as the end-to-end scheme), and the smallest segment could be one hop (i.e. hop-by-hop). In other words, OM-FEC is an adaptive strategy that tunes the architectural complexity between the extremes of end-to-end and hop-by-hop operation.

7.2.4 Feasibility of Intermediate FEC Coding/Decoding

In OM-FEC, since intermediate nodes perform FEC encoding/decoding, we would like to evaluate the limitations of applying FEC codes in video streaming and the feasibility of the decoding/ coding computation at the intermediate nodes. The RS(n, k) encoder takes k data packets and generates $n-k$ parity packets. Given the position of the lost packets, the RS decoder can reconstruct up to $n-k$ lost packets out of a total of n packets. Hence, a larger ratio n/k leads to a higher level of protection for the video data. In a video streaming system, n cannot be chosen arbitrarily large, since the video data are time sensitive. Larger values of n imply longer delays at the receiver. The maximum value of n is related to the bitrate of the encoded video, its packet size, and the buffering time at the receiver side. Let the FEC encoded bitrate be β bps, the packet size η bytes, and the receiver buffer size λ seconds. If the receiver buffer is full, the total amount of bits in the buffer is $\lambda\beta$. The amount of bits in a network packet is 8η, so the total amount of packets in buffer is $\lambda\beta/8\eta$. If RS(n, k) is deployed over k packets, the receiver needs to be able to accommodate at least n packets at the receiver buffer. Therefore, we need $n \leq \lambda\beta/8\eta$.

Using a systematic code, the encoder picks groups of k source data symbols to generate $n-k$ parity symbols. Every source data symbol is used $n-k$ times, so we can expect the encoding time to be a linear or approximately linear function of $n-k$. Since our system relies on real-time FEC encoding/decoding, it is necessary to evaluate the performance of the RS codec. We tested our implementation of the RS codec (based on Phil Karn's RS codec [81]) on a Dell PC with Pentium 4 CPU at 2.0 GHz, with 256 MB RAM, running Linux RedHat 8.2, with $n = 255$, and k

variable. The time needed to produce $n-k$ parity packets, given k data packets, is shown (in ms) in Table 2, for various values of $n-k$ and packet size

$n-k$	5	10	15	20	25	30	35	40
256 bytes/pkt	1.1	1.9	2.2	3.3	3.9	4.3	4.9	5.4
512 bytes/pkt	2.0	3.7	4.3	6.5	7.8	8.6	9.8	10.8
1024 bytes/pkt	4.2	7.3	8.6	13.0	15.6	17.2	19.5	21.6

Table 7.4: RS encoding time (in ms) as a function of $n-k$ and packet size

From Table 7.4, we observe that very high FEC encoding rates can be achieved even on commodity PCs (which would most likely be the peers of the overlay network). For example, the encoding bitrate of the $RS(255, 245)$ code can be up to 274 Mbps at packet size 1024 bytes, and this code can recover lost packets at random loss rates up to 3.92%. Erasure codes tested in [82, 69] gave similar results.

7.3 Results

We now demonstrate the effectiveness of OM-FEC by comparing it with end-to-end based FEC and hop-by-hop based FEC in both simulations and controlled real Planet-lab network [10] experiments. Based on the OM-FEC algorithm, if the available end-to-end bandwidth is large enough for both video data and FEC, then OM-FEC works in the end-to-end mode. In case of severe network congestion where none of the hops have enough bandwidth for both video data and FEC, OM-FEC partitions the overlay path into the smallest segments (i.e. hop-by-hop) and works the same as does hop-by-hop FEC. In this scenario, both hop-by-hop FEC and OM-FEC outperform end-to-end FEC in terms of video quality. Here, we focus our simulations and experiments on the cases that lie between the above two extremes. We first test the bandwidth efficiency of OM-FEC vs. the end-to-end scheme. This is followed by video simulations that compare the performance of OM-FEC against both end-to-end and hop-by-hop FEC. Finally, we perform a controlled Planet-lab network [83] video experiment. We expect that, as the number of hops increases and the variation of their loss rates becomes larger, OM-FEC will outperform the end-to-end FEC scheme. This expectation is confirmed by our simulations and

experiments. In this section, all the plotted curves are the averages of at least 10 simulation (experimental) runs.

7.3.1 Simulations - Bandwidth Efficiency

In this section we compare OM-FEC and end-to-end FEC in terms of their provided video throughput. We assume the task is to transmit a video encoded at 512 Kbps (which also represents the highest possible video throughput) through a sequence of overlay nodes. The simulation configuration is shown in Fig. 7.6. The topology includes one sender, one receiver, and three intermediate overlay nodes, with $L1$ through $L4$ denoting the overlay hops or links. Similar to [84] and [85], we use the two-state Markov (Gilbert) model to simulate packet loss on each hop. The sender puts out video packets through the three overlay nodes to the receiver, and the feedback information is provided via the same nodes but in the reverse direction. The probing packet is sent from the server once every 100 ms. We begin

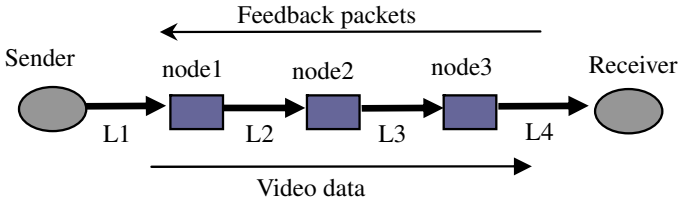


Figure 7.6: Simulation configuration for bandwidth efficiency; we vary the loss rates on each hop L1-L4 and compare the video throughput of OM-FEC vs. the end-to-end scheme.

our simulation by starting the network in a state of slight congestion. The simulation parameters are shown in Table 7.5. The average burst lengths are in the range of 2-3 packets. We set the RTT and range of packet loss rates for each hop. The network condition is changed every 300 ms. At time $t = 0$, the sender begins to puts out video data to the receiver. Information gathered by the probe packet from each hop is fed back to the sender. For the Basic Test, the sender calculates the available bandwidth of each link, as shown in Fig.7.8, according to the measured loss rate (Fig. 7.7) and RTT on each hop. The sender determines what kind of FEC scheme should be deployed for the current network condition. OM-FEC uses bandwidth more efficiently in case of network congestion as shown in Fig. 7.9, where

Test	Basic Test		Test A		Test B	
	Loss rate	RTT (ms)	Loss rate	RTT (ms)	Loss rate	RTT (ms)
L1	[1-2%]	10	[1-2%]	10	[2-3%]	10
L2	[1-4%]	30	[3-5%]	30	[3-6%]	30
L3	[3-5%]	10	[3-5%]	10	[3-6%]	10
L4	[2-4%]	20	[2-4%]	20	[3-4%]	20
RS(n,k)	$k = 80$, $\varepsilon = 0$, n is variable					
Network	conditions change every 300 ms					
Video	Encoded video bitrate = 512 Kbps					

Table 7.5: Simulation parameters for three different tests: Basic Test, Test A, and Test B. Loss rates are randomly chosen from their defined range.

video throughput is defined as bandwidth occupied by the video data. To test our approach in a heavier congestion condition, we increase the loss rate of the links in the simulation setup shown in Table 7.5, for Test A and Test B. In Fig. 7.10, we can see that OM-FEC outperforms the end-to-end scheme by a large margin at severe congestion. For end-to-end FEC, the increased end-to-end loss rate results in more FEC overhead over the entire overlay path. On the other hand, OM-FEC only considers the related segments where loss rate increases. Thus, the OM-FEC scheme has better performance than the end-to-end FEC at severe congestion in terms of bandwidth utilization.

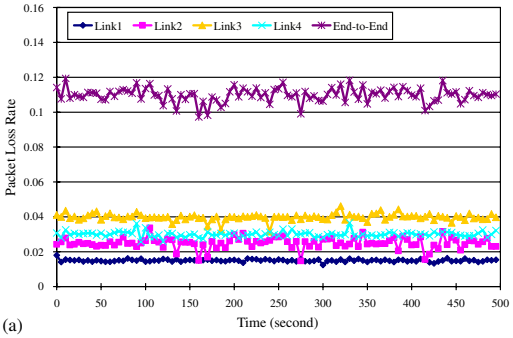


Figure 7.7: Packet loss rate on the overlay path

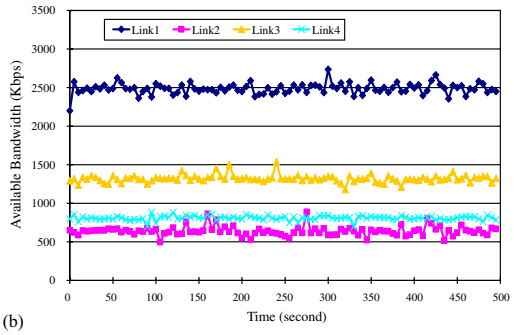


Figure 7.8: Available bandwidth of each hop

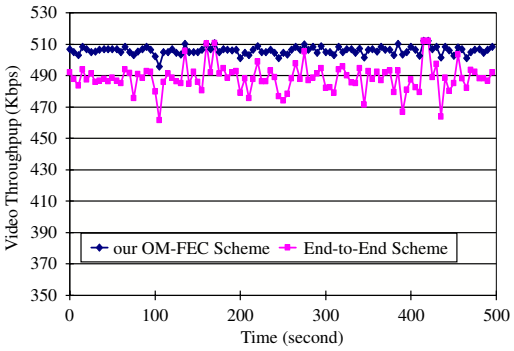


Figure 7.9: Video throughput of OM-FEC vs. end-to-end scheme.

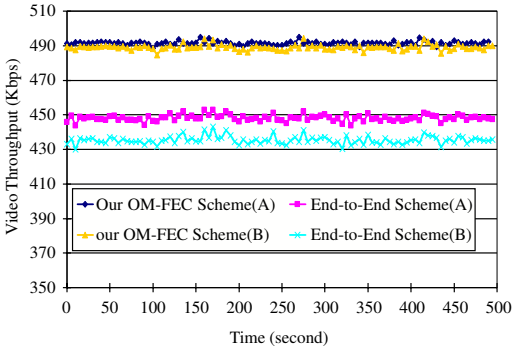


Figure 7.10: Video throughput of OM-FEC vs. end-to-end scheme in Tests A & B.

7.3.2 Video Simulations

In this part, we use a more complex overlay path to compare the performance of OM-FEC vs. end-to-end and hop-by-hop FEC. The path is shown in Fig. 7.11, where 10 hops are denoted $L1 - L10$. The packet loss rate for each hop is randomly chosen in the range [0.4%, 1.2%], thus the overall path loss rate is approximately in the range [4%, 12%]. The video sequence is Foreman, CIF resolution, 30 frames per second. The video bitstream is encoded using an H.263+ encoder with error-resilient option at 1530Kbps, with intraframe refresh at every second. For simplicity, the available bandwidth of each hop is fixed at 1656Kbps, therefore a maximum 126 Kbps bandwidth can be allocated to FEC, which can recover lost data up to a 7.6% packet loss rate. The network packet size is 512 bytes. Regarding the choice of RS(n,k) FEC codes, we fix $k = 85$, while n is determined by the loss rate and the available bandwidth. Based on our test, we got good FEC performance with $\varepsilon = 3$ and $\zeta = 6$, respectively. The network conditions change every 300 ms, and the probe interval is set to 100 ms.

Based on this setup, the server sends out a video bitstream using OM-FEC, hop-by-hop based FEC, and end-to-end based FEC. The results are shown in Fig. 7.12. Since each hop has a packet loss rate only between 0.4% and 1.2%, the hop-by-hop based FEC scheme has enough bandwidth for FEC to recover almost all the packet losses except for a very few observed burst losses. On the other hand, end-to-end FEC deploys FEC based on the end-to-end loss rate which is approxi-

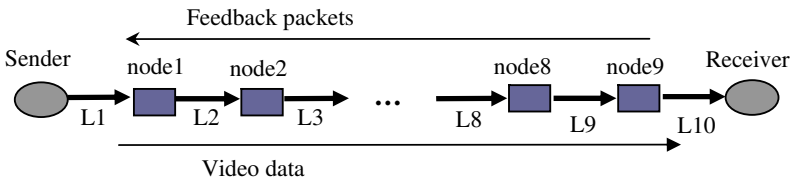


Figure 7.11: Video simulation path configuration, with varying loss rate on each hop L1-L10. We compare the performance of OM-FEC vs. end-to-end and hop-by-hop.

mately ranging in [4%, 12%]. However, due to the limited available bandwidth, the maximum end-to-end FEC scheme can only recover a loss rate under 7.6%. Therefore, we observed very bad video quality on the end-to-end based FEC scheme. Our proposed OM-FEC partitions the overall path into segments based on the available bandwidth and loss rate in order to acquire essentially the same video quality as hop-by-hop based FEC, with less computational complexity. OM-FEC partitions the overlay path in such a way that the added FEC codes can almost always fully recover the packet loss of each segment. We observed similar video results to those of hop-by-hop based FEC. The drops in the peak signal-to-noise ratio (PSNR) curve in OM-FEC are due to burst losses.

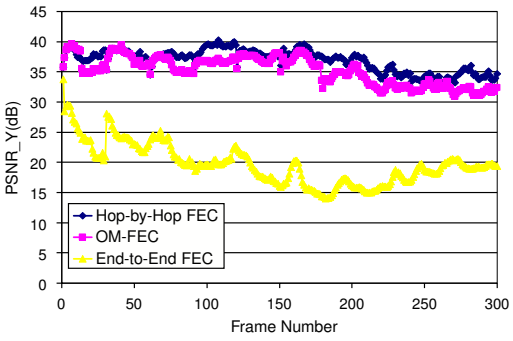


Figure 7.12: OM-FEC vs. hop-by-hop and end-to-end FEC.

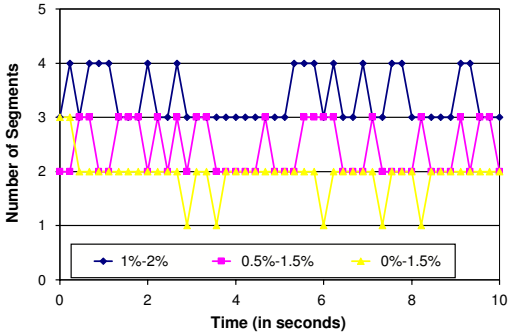


Figure 7.13: Several sample partitionings of OM-FEC.

In Fig. 7.13, we show several partition samples of OM-FEC at different per-hop loss rates. Though OMFEC uses fewer nodes to do the FEC computations, it can achieve similar video performance to that of hop-by-hop FEC. In Table 7.6, we compare the computational complexity of OM-FEC with hop-by-hop and end-to-

Per-hop loss range (%)	Approx. total loss rate(%)	Number of OM-FEC segment	Avg. OM-FEC segments (in 10 runs)	Number of hop-by-hop segments	Number of end-to-end segment
0.0-1.5	0.0-15	1-3	1.96	10	1
0.4-1.2	4-12	1-2	1.99	10	1
0.5-1.5	5-15	2-3	2.35	10	1
1-2	10-20	2-4	3.32	10	1

Table 7.6: Computational complexity comparison of the three FEC schemes

Server	nima.eecs.berkeley.edu
Node2	planetlab-1.cmcl.cs.cmu.edu
Node3	planetlab1.cs.cornell.edu
Receiver	video.testbed.ecse.rpi.edu

Table 7.7: Nodes involved in Planet-Lab experiments

end. Obviously, the number of segments in the end-to-end scheme is always 1 (no intermediate node is involved in FEC computation) and the number of segments in hop-by-hop is always 10 (each overlay node does FEC decoding/re-encoding). On the other hand, OM-FEC partitions the overlay path based on the loss rate and the available bandwidth. It uses fewer nodes than hop-by-hop based, but with nearly the same performance in terms of video quality at the receiver side.

7.3.3 Controlled Planet-Lab Network Experiments

We also implemented our protocol over the real Internet using the Planet-Lab infrastructure [83]. The implementation includes an overlay agent and the protocol itself. Our overlay agent can run on any Linux Planet-Lab node. Each agent forwards a video packet to the next node until it arrives at the destination. The experimental topology is the same as in Fig. 7.5 and the Planet-Lab nodes involved are listed in Table 7.7. In the experiments described in this section, we measure the objective video quality at the receiver in terms of the PSNR. We use the same video sequence as in Section 7.2.2 except that the resolution is QCIF and encoded bitrate is now 512Kbps. Since there is virtually no congestion from UC Berkeley to RPI (Internet 2), packets are artificially dropped to simulate a

congestion effect. A similar approach has been considered by Nguyen and Zakhor in [71], where the authors artificially drop packets to simulate 10% loss rate from Sweden and Indiana to UC Berkeley. The packet loss rate from Utah to CMU is set to 5%, other links are set to 1% [84, 86]. The actual packet loss, from Utah to CMU simulation, was 5% with a standard deviation of 1.1%. For the other links with 1% loss, we just randomly dropped 1 packet out of each 100. The available bandwidth from Utah to CMU is also upper bounded to 550 Kbps. Under these conditions, the end-to-end scheme designs an FEC code based on the 550 Kbps bandwidth and total loss rate 8%. Here, we set $\varepsilon = \zeta = 0$. OM-FEC identifies the bottleneck and partitions the overlay into three segments as follows: segment 1 from Server to Node 1, segment 2 from Node 1 to Node 2, and segment 3 from Node 2 to the receiver. Two nodes are involved in FEC encoding/decoding. The FEC is deployed within each segment. OM-FEC places FEC at the bottleneck for a bandwidth of 550 Kbps and 5% loss rate. It can recover more packet loss than the end-to-end scheme and its video quality is much higher than that provided by the end-to-end scheme FEC, as shown in Fig. 7.10. The PSNR gains are on the order of 13 dB.

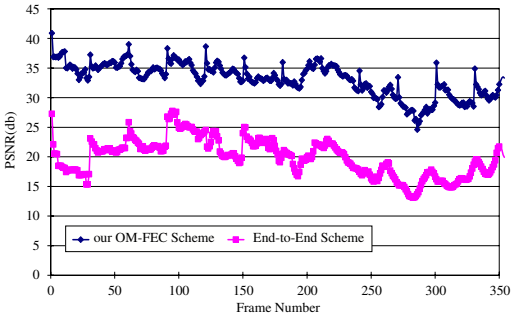


Figure 7.14: Video PSNR of OM-FEC vs. end-to-end FEC (four hops).

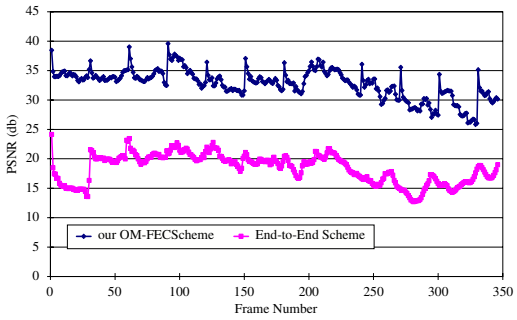


Figure 7.15: Video PSNR of OM-FEC vs. end-to-end FEC (five hops).

We add one overlay node (Node 4: planet1.ecse.rpi.edu) to the path at last hop with 1% loss rate in the second set of experiments. In this case, for the end-to-end scheme, the FEC is designed based on the bandwidth of 550 Kbps and loss rate of 9%. OM-FEC still partitions the overlay path into segments as before, and the FEC at the bottleneck is still designed for the bandwidth of 550 Kbps and 5% loss rate. Still, two nodes are involved in FEC encoding/decoding. The PSNR

results are contained in Fig. 7.15, which shows that the advantage of OM-FEC over end-to-end FEC is increased compared to Fig. 7.14. Here, the PSNR gains are on the order of 14 dB. As the number of nodes involved in the transmission increases, OMFEC performs dramatically better than the end-to-end scheme. For visual comparison, we show a few decoded frames in Figs. 7.16 and 7.17. These high PSNR improvement figures occurred because we have tried to send at a high video bitrate, relative to what is available on the links. Of course, if we had tried to send at a lower rate, there would be less difference between the various results (see Figs.7.14 to 7.17).

Figure 7.16: Video streaming over four hops: OM-FEC (left) vs. end-to-end FEC (right).

7.4 Conclusions

We have proposed an OM-FEC approach for streaming video over P2P networks, which automatically adapts its architectural complexity between the ex-

Figure 7.17: Video streaming over five hops: OM-FEC (left) vs. end-to-end FEC (right).

tremes of pure end-to-end or pure hop-by-hop operation. The proposed OM-FEC improves the video throughput of the constructed peer-based overlay transmission path by dividing the overlay path into segments based on link characteristics, and applying the appropriate amount of FEC over each segment. We have shown that video streaming using our approach outperforms that of end-to-end FEC without incurring high per-hop complexity.

CHAPTER 8
Distributed FGA-FEC

In the previous chapter, we proposed an OM-FEC method to efficiently utilize one congested overlay path. In this chapter, we investigate a distributed FGA-FEC scheme over a congested multihop network, where we do FGA-FEC decode/recode at selected intermediate overlay nodes, and do FGA-FEC adaptation at remaining nodes. In order to reduce the overall computational burden, we propose two methods: (1) a coordination between optimization processes running at adjacent nodes to reduce the optimization computation, and (2) extension of OM-FEC from Chapter 7 to reduce the number of FGA-FEC decode/recode nodes. Simulations show that the proposed scheme can greatly reduce computation, and can provide near best possible video quality to users.

8.1 Motivation

In Chapters 3 and 4, we proposed FGA-FEC for encoding and adaptation of scalable video to simultaneously serve diverse users. We assumed that there was no congestion in the network backbone, i.e. that the backbone available bandwidth was large enough to accommodate all user requirements. Therefore, the server first encoded the scalable video based on the highest user request and aggregated network conditions, then it sent the encoded bitstream into the network. Inside the network, the DSNs adapted the FGA-FEC encoded bitstream to satisfy heterogeneous users by shortening and/or dropping packets. One problem still remained: congestion could be anywhere inside the network, especially in a multihop *ad hoc* wireless network. How should we modify FGA-FEC to work with a congested backbone? Here, a congested link is defined as a link whose available bandwidth is less than the minimum required bandwidth to accommodate a user's video request. One solution to address this problem is to optimize FEC protection for each individual link and apply FGA-FEC decode/recode at each DSN for each user. By FGA-FEC decode/recode, we mean that a DSN decodes FGA-FEC of the received GOP, re-

optimize the multiple descriptions and then re-codes the GOP with new designed FGA-FEC for its downlinks. This would be a heavyweight hop-by-hop computationally intensive method if done at every overlay node. Here, we argue it may not be necessary to do FEC decode/recode at each DSN. For example, in Fig. 8.1, if the link between the server and the DSN1 is congested, and other links are not, we may only do FGA-FEC decode/recode at DSN1 and do FGA-FEC adaptation at the remaining DSNs. We need to identify the congested links in the backbone and apply the appropriate transformation at each DSN. Still, running the full FGA-FEC optimization at even some DSN nodes may be computationally demanding. So, here we describe a *distributed algorithm*, where we do FGA-FEC decode/recode at the selected DSNs. The proposed distributed FGA-FEC scheme includes two parts: (1) a coordination method between FGA-FEC optimization processes running at nearby nodes to reduce the optimization computation, and (2) we apply OM-FEC to reduce the number of FGA-FEC decode/recode nodes, i.e we use FGA-FEC adaptation where permitted and perform FGA-FEC decode/recode only at certain key DSNs. This design thus lies between the end-to-end and hop-by-hop paradigms. If there is no congestion over the backbone, we choose end-to-end FGA-FEC scheme, no FEC decode/recode is needed at intermediate nodes. If each backbone link is congested, it is a heavyweight hop-by-hop FEC decode/recode scheme. For this more advanced distributed algorithm, we only focus on SNR scalability, and leave extension to resolution and frame-rate scalability as a topic of future work.

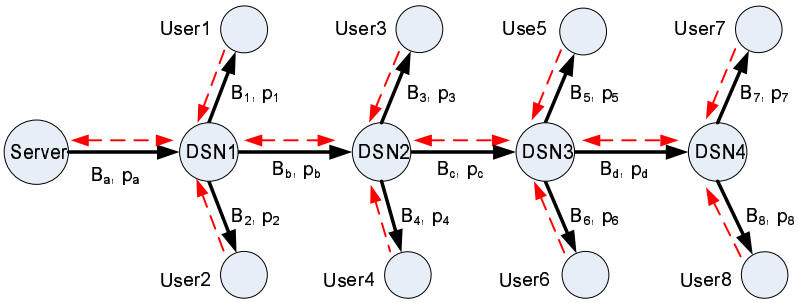


Figure 8.1: Streaming video from server to users through DSNs, red-dotted arrows are overhead information flows, black-solid arrows are video flows.

We outline our idea in a simplified example as shown in Fig. 8.1, where a server streams video to 8 diverse users through DSNs over a congested backbone. Before

the streaming session, each end user sends its ideal video request and maximum tolerable distortion to its directly connected DSN. During the streaming, at each time interval (1 GOP or multiple GOPs), edge DSNs (DSN4, whose downlinks have only end users) initialize optimization processes for each child to figure out what kind of bitstream it needs to request from its parent DSN (DSN3). This request is based on its children's link conditions and their video requests. The combined video request of its child nodes along with the optimization result is sent to DSN3 as overhead information. DSN3 then runs optimizations for its own children, including DSN4 (DSN3 treats DSN4 as one ordinary user), and generates the requested information to its parent DSN2. This process is repeated until we arrive back at the server. The server then runs the same algorithms as DSNs to determine the amount of FEC that should be applied to the video and then sends the encoded video into network. Inside the network, some selected DSNs will decode, redo the FGA-FEC design and recode FEC for some users. There are two kinds of flows in the distributed algorithm, upstream overhead information flow (shown via red-dotted arrows at Fig. 8.1) and downstream video data flow (shown via black arrows). Each DSN only exchanges optimization information with its direct parent or children, generating only local overhead information traffic. The DSNs use this information to coordinate optimization processes running at nearby nodes to reduce the computational burden, as well as to decide which nodes that will be involved in the FGA-FEC decode/recode. We apply the idea of OM-FEC to minimize the number of involved FGA-FEC decode/recode nodes while still maintaining the near optimal video quality.

8.2 Distributed FGA-FEC

The distributed FGA-FEC algorithm includes two parts: (a) a coordination between adjacent nodes to reduce the optimization computation, and (b) the idea of OM-FEC to identify congested links and select key nodes to do FGA-FEC decode/recode. We will first focus on how to reduce the optimization computation.

8.2.1 Coordination Between Algorithm Processes Running at Adjacent Nodes

As we know from Chapters 3 and 4, the optimization algorithm is run at both DSNs and video server. A DSN runs optimization for its children to figure out what kind of bitstream it needs to request from its parent DSN or server. The server runs optimization to design the FEC and to encode a GOP. The only difference in the optimization algorithms running at DSNs and server are the input parameters. The optimization time interval is one GOP. Here, we briefly overview the optimization algorithm.

The optimization goal is to find the optimal bitrate partition $R = \{R_1, R_2, \cdots, R_N\}$ of a GOP, which minimizes the end-to-end mean distortion $E[D(R)]$ over a channel with available bandwidth B and packet loss probability p.

$$E[D(R)] = \sum_{i=0}^{N} q_i D(R_i), \tag{8.1}$$

subject to:

$$\begin{cases} 0 \leq R_1 \leq R_2 \leq ... \leq R_N \\ R_{\text{total}} \leq B \\ R_i - R_{i-1} = r_i \times i, \qquad r_i \geq 0, \quad \forall\ i \in [1, N] \end{cases}$$

where N is the number of descriptions encoded in one GOP, r_i is the rate of each subsection at section i ($i \in [1, N]$) of the bitstream (please refer to Chapter 3 for definition of sections and subsections). The probability that any i out of N packets are successfully delivered is q_i, R_{total} is the total bandwidth (bitrate) available for both FEC and video data.

Solving (8.1) is a constrained optimization problem. To find the optimal solution, we can use the Lagrange multiplier method and construct the function

$$F(R_1, \cdots, R_2, \lambda) = \sum_{i=0}^{N} q_i D(R_i) + \lambda(\sum_{i=0}^{N} \alpha_i R_i - B). \tag{8.2}$$

Taking the partial derivative of (8.2) with respect to R_i, $i = 0, 1, \cdots, N$, and setting them to 0, then, we can use a bisection search to find the appropriate λ and the

corresponding rate break points $R = \{R_1, R_2, \cdots, R_N\}$, and the $E[D]$ value.

The motivation of coordination typically is from the following: (1) Video statistic information between adjacent GOPs does not change rapidly. (2) Server and parent DSNs have the optimization information from their child DSNs of the same GOP, with only different B and p. Therefore, the problem can be simplified into how to utilize the previous optimization information as network condition and video statistics change. We will use two coordination methods: (1) search with previous GOP results at this DSN, and (2) search with current GOP result from child node. Edge DSNs (a DSN whose children are all end users) initialize optimization for a new GOP. There, we can use optimization information from the previous GOP, we call this method "search with previous GOP". Intermediate DSNs and the server have local information not only of the same GOP from child DSNs but also have their previous GOP optimization result. Thus, they can use information of either of these GOPs to initialize their optimization search. Using the optimization information from child DSN, will be called "search with neighbor". We also consider a full search method, where each node runs the optimization algorithm independently. There, the upstream communication between nodes is only the video request.

8.2.2 Coordination to Reduce Number of FGA-FEC Decode/recode Nodes

An extreme case of the distributed FGA-FEC is hop-by-hop FGA-FEC decode/recode, i.e do FGA-FEC decode/recode at each DSN. This method can provide the best possible video quality for diverse users in a congested backbone, since the protection is specifically optimized for each individual user. One may argue that it is not necessary to do the FGA-FEC decode/recode at each DSN, if only part of the network is congested. For example, we already have shown that if the network backbone is not congested, our simpler FGA-FEC adaptation can also provide a near optimal solution if the user diversity is not too great. Combining these two ideas together, we do FGA-FEC decode/recode at some selected nodes, while still providing similar video quality to hop-by-hop FGA-FEC decode/recode. So here, we apply the OM-FEC concept to the network backbone to divide the network into

segments and hence minimize the number of FGA-FEC decode/recode nodes.

We use the topology of Fig. 8.1 to illustrate the idea. In Fig. 8.1, if there is no congestion in the backbone, we can directly encode a video using FGA-FEC only at the server and then use the simpler FGA-FEC adaptation inside the network. If some links in the backbone are congested, we need to identify them and apply FGA-FEC decode/recode functions at the edge nodes of these congest links. We still use local information to decide the congested links. Algorithm 7 works as follows.

Algorithm 7: Distributed FGA-FEC Algorithm

1. Edge DSN initializes new GOP optimization for its children using "search with previous GOP".
2. Edge DSN sends video requests and the optimization information $\{D, D_{\max}, \lambda B, p, p_{\max}\}$ to its parent DSN.
3. Intermediate DSN runs optimization for its children using "search with neighbor".
4. Intermediate DSN tests if its child DSN has enough bandwidth B for an FGA-FEC adaptation.
   If yes, intermediate DSN informs its child DSN as an adaptation node.
   Otherwise, this is a congested link, child DSN will do FGA-FEC decode/recode.
5. Intermediate DSN sends video request and optimization information to its parents.
6. This process is repeated through intermediate DSNs back up to server. Thus, the congested links are detected and so are the FGA-FEC decode/recode nodes.

Referring to Fig. 8.1, the network conditions are listed for each link. Before the streaming session, each end user sends its video requirement to its parent DSN as overhead information. This requirement can be described as a quality range $[D_{\min}, D_{\max}]$, where $D_{\min}$ is the user's ideal video preference and $D_{\max}$ is his/her maximum tolerable distortion. The optimization starts from the edge DSN (DSN4) and proceeds up to the server using a bottom up procedure through all the intermediate DSNs.

1. DSN4 runs optimization algorithm for its child User7, the result will be $D_{\min 7}+\lambda_7 R_{\text{total}7}$, given $B_7 \geq R_{\text{total}7}$, we can only use part of the available bandwidth, since $D_{\min 7}$ is satisfied. If $B_7 < R_{\text{total}7}$, link is congested, and the result will be $D'_7 + \lambda'_7 R'_{\text{total}7}$, where $D_{\min 7} < D'_7$ and $R'_{\text{total}7} = B_7$, i.e. we use up all the available

bandwidth to get the best quality possible for User7. If $D_{\max 7} < D'_7$, no video is sent to User7.

Similarly for User8, the result will be $D_{\min 8} + \lambda_8 R_{\text{total}8}$ given $B_8 \geq R_{\text{total}8}$. If $B_8 < R_{\text{total}8}$ the result will be $D'_8 + \lambda'_8 R'_{\text{total}8}$, where $D_{\min 8} < D'_8$, and $R'_{\text{total}8} = B_8$.

2. DSN4 generates its video request to send upstream which is the union of both users' request, in this case, it is D =min($D_{\min 7}$ or D'_7, $D_{\min 8}$ or D'_8), given both users have valid data request. The maximum tolerable distortion will be $D_{\max}$ =max($D_{\max 7}, D_{\max 8}$) (at least should satisfy one user), the requested bit-stream range will be at [D, $D_{\max}$]. DSN4 sends this combined video request to its parent DSN3, along with its optimization information $\{\lambda, B, p\}$. Here, (B, p) corresponds to D. The total request is the set $\{D, D_{\max}, \lambda, B, p, p_{\max}\}$, where $p_{\max}$ is the maximum loss probability among its children links and is used for FGA-FEC optimization test.

3. DSN3 runs the optimization algorithm for its children (including DSN4 with video request [D, $D_{\max}$] and network conditions (B_d, p_d), based on the optimization information λ from DSN4 (DSN3 uses the optimization information to start its own process).

4. DSN3 does one FGA-FEC optimization test, it optimizes FGA-FEC based on B_d and the aggregated loss probability, roughly $1-(1-p_d)(1-p_{\max})$. If B_d is large enough for FGA-FEC encoding (can provide the video quality D with FGA-FEC), DSN4 is notified as an FGA-FEC adaptation node, otherwise, the link between DSN3 and DSN4 is congested, both DSN3 and DSN4 need to do FEC decode/recode.

5. DSN3 sends its own request $\{D, D_{\max}, \lambda, B, p, p_{\max}\}$ to DSN2.

6. This process is repeated upward to server through DSN2 and DSN1. The backbone is then divided into segments and appropriate nodes are selected for FEC decode/recode. Similarly as OM-FEC, the partition result could be an end-to-end scheme (no intermediate FEC decode/recode is performed, only FGA-FEC adaptation) if no congestion is detected in the backbone, or could be a heavy weight hop-by-hop FEC decode/recode scheme, given each hop is congested in the backbone.

7. The server runs the optimization based on video request and optimization

information from DSN1, encodes the GOP with FGA-FEC, and sends it to DSN1.

We test a link using FGA-FEC optimization to see if the link available bandwidth is large enough for a downlink FGA-FEC adaptation. If not, the link is congested. For a congested link, there is a possibility that the bandwidth is large enough for a hop-by-hop FEC decode/recode (the optimization result can satisfy the video request only for this hop), but can not satisfy an FGA-FEC adaptation. In this situation, we need to explore how to efficiently utilize the link by assigning appropriate FEC for this link. The solution is that we will use more bandwidth and add more protection to the optimized FEC in the server side. We show this in an example as in Fig. 8.2 where the link available bandwidth between server and DSN is large enough for FEC decode/recode at DSN but not enough for FGA-FEC adaptation.

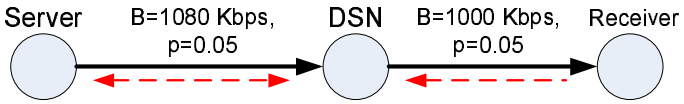


Figure 8.2: A simple topology streaming video to a user through DSN, the channel condition is listed at each link, this backbone is congested for FGA-FEC, but not congested for FEC decode/recode.

While DSN does the optimization, it assumes that the available video source bitstream is large enough to fulfill the receiver's needs. Actually, this is not always true, there is possibility that the received video at DSN is not enough for receiver. Since there is more bandwidth available, we can add additional protection to the optimized FEC (for FEC decode/recode) at server side as we did at OM-FEC scheme. We use the Fig. 8.2 topology to analyze the effect of adding more protection over the assigned FEC, the video bitstream is *Foreman* GOP1, N =64.

(1) DSN first does the optimization based on user's channel condition, the resulting rate break points are $R_0 = R_1 = \cdots = R_{54} = 0$ Kbps, $R_{55} = 156$ Kbps, $R_{56} = 415$ Kbps, $R_{57} = R_{58} = \cdots = R_{64} = 880$ Kbps, RS(N, i) is applied to section i. The expected distortion at receiver is 38.15 dB. The DSN needs video source bitstream up to bitrate 880 Kbps delivered to it, then it can satisfy the user's request, i.e at least 57 out of 64 packets should be delivered to DSN.

(2) The video request is sent to server, since there is enough bandwidth be-

tween server and DSN, the server uses the same protection as DSN optimization to satisfy the $E[D]$ =38.15 dB, and encodes 64 packets.

(3) While streaming over the channel, the probability of receiving at least 57 packets at DSN is 0.9597, i.e. there is a probability of 0.0403 that the DSN can not get enough source video to fulfill its bitstream requirement.

(4) Since there is enough bandwidth for server, now we add more protection to the optimized FEC protection. We consider directly add a number of FEC packets to the original 64 descriptions, we add an upper integer of $\epsilon = \sqrt{Np(1-p)}$ parity packets which is the standard deviation of q_i (the probability of i out of N packet successfully received). In this specific test, $\epsilon = \sqrt{64 \times 0.05 \times 0.95} = 2$. i.e. we are protecting section i with RS$(N+\epsilon, i)$. After more protection added to the optimized results, the probability of receiving at least 57 packets at DSN becomes 0.9945, i. e. there is a probability of only 0.0055 that the DSN can not get enough source video to fulfill its bitstream requirement. Since the optimization is based on B=1000 Kbps, N=64, after adding two more FEC packets, the total bandwidth consumed in the backbone is $66 \div 64 \times 1000 = 1031$ Kbps, still less than 1080 Kbps. This can be explain by the Fig. 8.3.

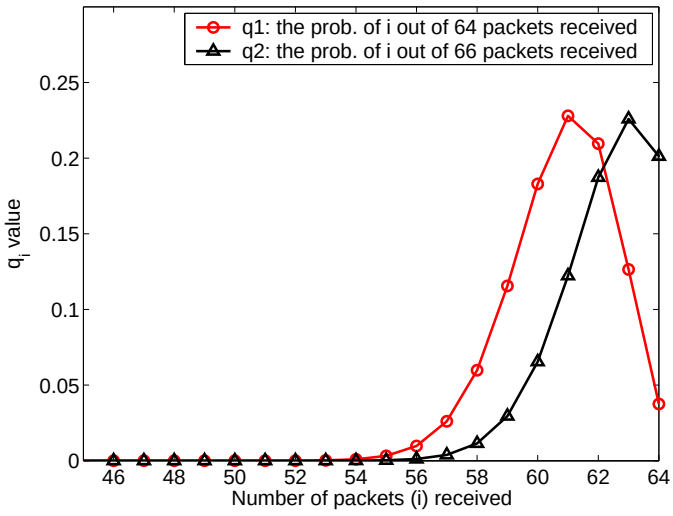


Figure 8.3: The probability of i out of N packets is successfully received at different protection, for q_2, we add additional 2 parity packets.

Add additional protection actually push the q_i curve to the right and hence decrease the probability of receiving less than a certain number of packets.

8.3 Experiments and Simulations

We did experiments and simulations to show the efficiency of our proposed distributed FGA-FEC scheme using videos *Foreman* CIF, 18 GOPs, *Mobile*, SIF, 8 GOPs and *Football*, SIF, 7 GOPs, with 16 frames/GOP in all three sequences. The source encoder is MC-EZBC, $N = 64$. The proposed scheme includes two approaches (1) a coordination method between optimization processes running at adjacent nodes to reduce computation, (2) using the OM-FEC concept to reduce the number of FGA-FEC decode/recode nodes while still maintain near optimal video quality, measured in terms of PSNR. Regarding the first approach, we compare the number of iterations need to reach the optimization stop point using "full search", "search with previous GOP" and "search with neighbor". For the later approach, we compare with hop-by-hop FEC decode/recode scheme and show that we can get similar video quality, but use fewer node involved in FEC decode/recode. Finally, we measured the CPU time of using the distributed FGA-FEC algorithm to show the efficiency.

8.3.1 Optimization Performance

We solve the optimization problem using a bisection search to find the best λ value. We need find a stopping criteria. We use $|R_{\text{total}} - B| < \frac{1}{N} \times B$ and $|\lambda - \lambda_{\text{previous}}| < \varepsilon$, i.e the total rate should be close to the available bandwidth and λ is not changing much, where ε is a threshold. Intuitively, a larger threshold should correspond to coarser precision. After the optimization, $(N - \frac{1}{N} \times B) < R_{\text{total}} < (N + \frac{1}{N} \times B)$. If $R_{\text{total}} < B$, we need to allocate more video data to R_N to satisfy $R_{\text{total}} = B$. If $R_{\text{total}} > B$, we need to remove some video data from R_N to satisfy $R_{\text{total}} = B$. We did experiments to show the effect of varying the ε threshold. Tables 8.1 and 8.2 show the quality loss (PSNR loss in dB), compared with solutions using threshold $\varepsilon = 1 \times 10^{-9}$ under various network conditions.

In Tables 8.1 and 8.2, larger thresholds correspond to coarser optimization precision. Still at $\varepsilon = 1 \times 10^{-5}$, the quality loss is almost negligible. So we will use threshold 1×10^{-5} in our experiments. The larger threshold can reduce the number of iterations needed. We evaluate this below.

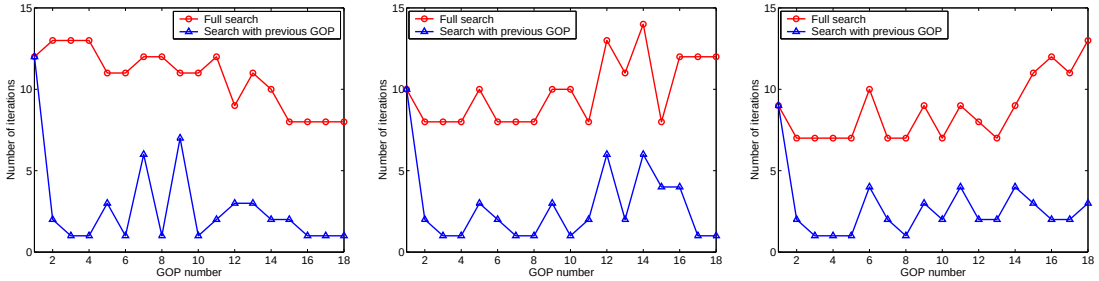
Channel	B=1200 Kbps, p=0.12			B=1200 Kbps, p=0.03		
Threshold	1×10^{-6}	1×10^{-5}	1×10^{-4}	1×10^{-6}	1×10^{-5}	1×10^{-4}
Foreman	7.2 E-6	3.6 E-3	1.3 E-1	7.8 E-6	1.4 E-3	8.5 E-2
Football	4.0 E-4	1.1 E-2	7.4 E-2	9.2 E-5	3.9 E-3	4.6 E-2
Mobile	5.5 E-5	6.5 E-3	7.7 E-2	1.3 E-5	1.4 E-3	5.1 E-2

Table 8.1: The PSNR loss in dB of using various stop search threshold, compared with solutions obtained by set a threshold to a very small value 1×10^{-9}.

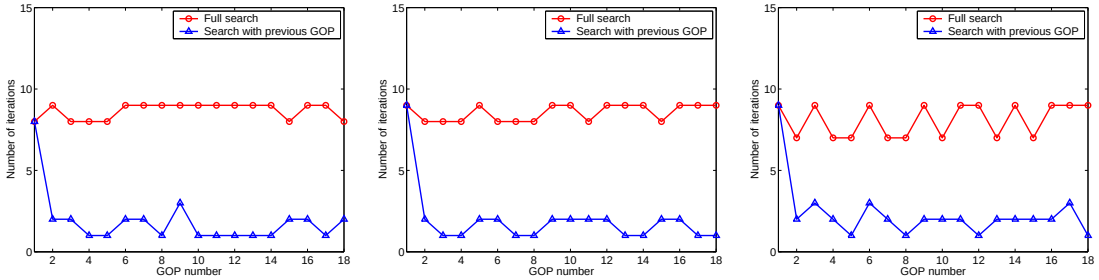
Channel	B=400 Kbps, p=0.12			B=400 Kbps, p=0.03		
Threshold	1×10^{-6}	1×10^{-5}	1×10^{-4}	1×10^{-6}	1×10^{-5}	1×10^{-4}
Foreman	0	2.8 E-4	2.0 E-2	1.5 E-6	1.6 E-4	1.7 E-2
Football	0	0	7.4 E-2	0	0	1.5 E-2
Mobile	0	1.8 E-4	7.7 E-2	0	0	0

Table 8.2: The PSNR loss in dB of using various stop search threshold, compared with solutions obtained by set a threshold to a very small value 1×10^{-9}.

At an edge DSN, there is only previous GOP optimization information available. The DSN can use this information as a starting point for current GOP optimization. In Fig. 8.4, we show several examples comparing our proposed "search with previous GOP" with the "full search" in terms of number of iterations needed to reach the optimization stoping point at various available bandwidths and different packet-loss probabilities. Here one iteration is defined as one λ step calculation. Initially, we set $\lambda = 1 \times 10^{-3}$ in the "full search" method. This value is picked based on experimental results, we tested the optimization in different network conditions and observed on possible biggest value of $\lambda = 3 \times 10^{-4}$ at $p = 0.05$, $B = 100$ Kbps, *Foreman*, GOP1. For full search optimization, the bisection search starts from the initial λ to the optimization stopping point. In the "search with previous GOP" method, the first GOP is the same as full search, we start from an initial λ value 1×10^{-3} and search to the optimization stopping point. After the first GOP, we use the previous GOP final λ (optimal point value) as our starting point and search to an appropriate direction based on the analysis in the previous section. We can quickly reach the optimization stopping point within several iterations. Of course, if we set a different initial value for λ, the iterations of full search algorithm would



(a) B= 1200 Kbps,p=0.03, Threshold=1×10^{-9} (b) B=1200 Kbps, p=0.12, Threshold=1×10^{-9} (c) B=600 Kbps, p=0.06, Threshold=1×10^{-9}



(d) B= 1200 Kbps,p=0.03, Threshold=1×10^{-5} (e) B=1200 Kbps, p=0.12, Threshold=1×10^{-5} (f) B=600 Kbps, p=0.06, Threshold=1×10^{-5}

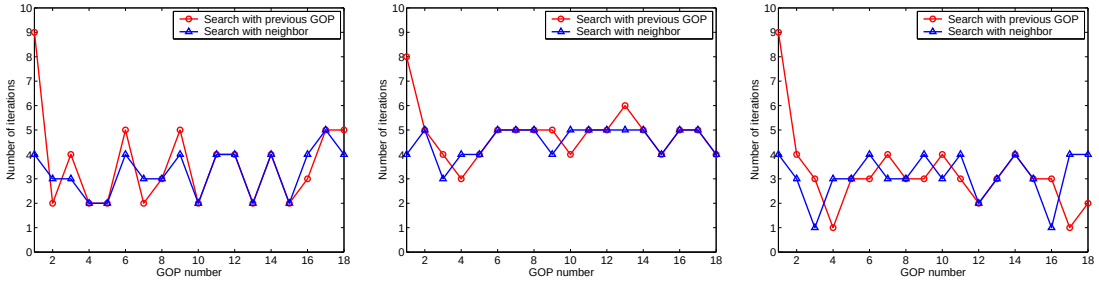
Figure 8.4: Effective Threshold: full search vs. search with previous GOP in terms of number of iterations vs. GOP number for different bandwidths and packet loss probabilities, (a-c) use threshold 1×10^{-9} and (d-f) use threshold 1×10^{-5}.

change. Table 8.3 shows the average number of iterations needed to reach the optimization stopping point in the tests in Fig. 8.4. As expected, larger threshold results in fewer number of iterations.

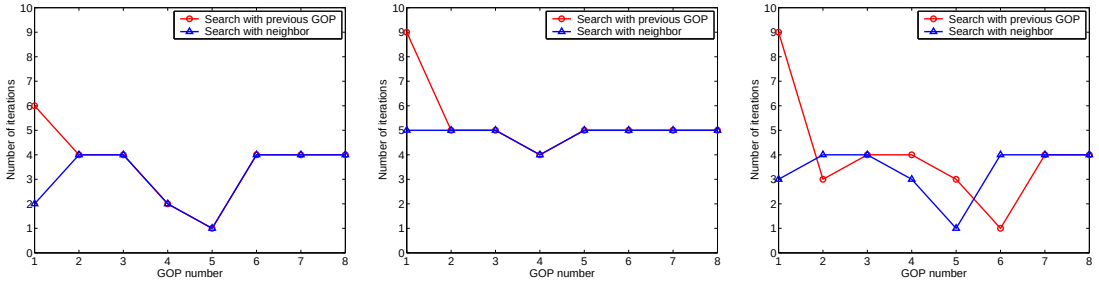
Threshold	1×10^{-9}		1×10^{-5}	
Search method	full	with previous GOP	full	with previous GOP
$B = 1200, p = 0.12$	9.89	2.83	8.56	1.94
$B = 1200, p = 0.03$	10.72	2.78	8.67	1.89
$B = 600, p = 0.06$	8.72	2.67	8.11	2.33

Table 8.3: The number of iterations to reach the optimization stopping point for various network conditions and search thresholds.

At intermediate DSNs and server, there is optimization information not only from previous GOP, but also overhead information from children DSNs about the same GOP. In Fig. 8.5, we compare "search with previous GOP" and "Search with



(a) *Foreman*, B= 1200, p=0.03 to B=600, p=0.06 (b) *Foreman*, B=400, p=0.12 to B=1200, p=0.03 (c) *Foreman*, B=600, p=0.06 to B=800, p=0.06

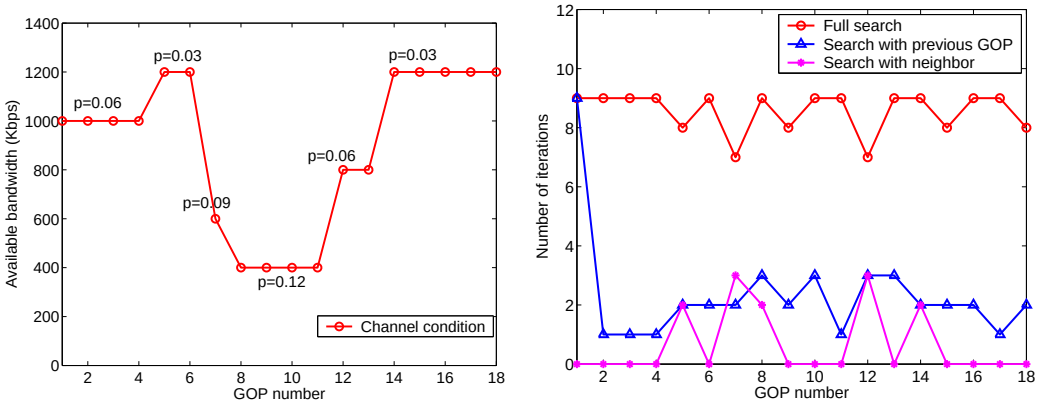


(d) *Mobile*, B=1200, p=0.03 to B=600, p=0.06 (e) *Mobile*, B=400, p=0.12 to B=1200, p=0.03 (f) *Mobile*, B=600, p=0.06 to B=800, p=0.06

Figure 8.5: Comparison of "Search with previous GOP" with "Search with neighbor" in terms of number of iterations to reach the optimization stopping point vs. GOP number. (a-c) *Foreman* and (d-f) *Mobile*

same GOP" methods, GOP by GOP. We show how to do the experiments in an example. For example, in Fig. 8.5(a) caption $B = 1200, p = 0.03$ to $B = 600, p = 0.06$, this means the previous information is obtained by optimizing a GOP with $B = 1200, p = 0.03$ and the current available network condition is $B = 600, p = 0.06$. In "search with previous GOP", we use information from previous GOP to optimize the current GOP for the current network condition. In "search with neighbor", we use the same GOP information in previous network conditions from child DSN. Here we process GOP by GOP and list them together in Fig. 8.5. Results show that if network condition changes, these two methods have similar performance.

In Fig. 8.6, we further compare the full search algorithm with our proposed "search with previous GOP" and "search with neighbor" methods on a dynamic channel, where the channel condition changes over the GOPs as in Fig. 8.6(a). The corresponding number of iterations to reach stopping points for the three methods are shown in Fig. 8.6(b). We can give an example to show how to do the exper-

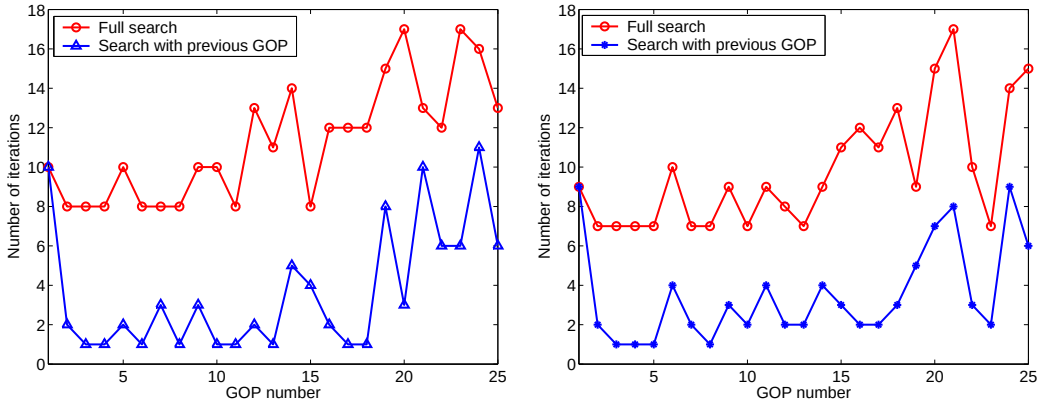


(a) Channel condition (b) Number of iterations

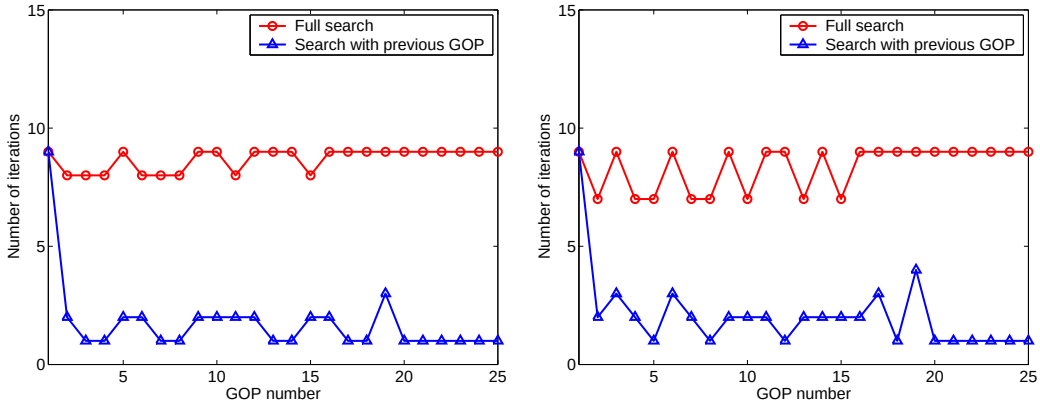
Figure 8.6: Dynamic Channel Conditions: full search algorithm vs. our our proposed "search with previous GOP" and "search with neighbor", in terms of number of iterations at a dynamic channel, (a) channel conditions varying over GOP number, (b) the number of iterations to reach optimal stopping point.

iment with "search with neighbor" method. For GOP 5, the child DSN first does optimization using $B = 1000$ Kbps, $p = 0.06$, and obtains a value λ, then uses this λ to optimize GOP5 in $B = 1200$ Kbps, $p = 0.03$ at parent DSN. If the network condition does not change, the optimization value can be used directly without optimization. From Fig. 8.6, we see that if the channel condition changes, both "search with previous GOP" and "search with neighbor" have similar performance, but when channel condition is statistically consistent, using "search with neighbor" gains over "search with previous GOP", saving about 2 iterations on average.

Fig.8.7 illustrate what happens at a scene cut during video streaming. The cut occurs at GOP 19, where the sequence changes from *Foreman* to *Football*. In both cases, at the scene cut, the number of iterations increase because the statistical character of these two clips are very different, hence the $D(R)$ curves are not similar at all. The results in this section show that with cooperation between adjacent DSNs, the optimization complexity is greatly reduced.



(a) B=1200 Kbps, p=0.12, Threshold=1×10^{-9} (b) B=600 Kbps, p=0.06, Threshold=1×10^{-9}



(c) B=1200 Kbps, p=0.12, Threshold=1×10^{-5} (d) B=600 Kbps, p=0.06, Threshold=1×10^{-5}

Figure 8.7: Comparison full search with our proposed "search with previous GOP" and "search with neighbor". There is a scene cut at GOP 19 from *Foreman* to *Football*. Both (a) and (b) use search threshold 1×10^{-9}, and (c) and (d), use 1×10^{-5}

8.3.2 Comparison of FGA-FEC Adaptation with Hop-by-hop FGA-FEC Decode/recode

Hop-by-hop FGA-FEC decode/recode is a special case of the distributed FGA-FEC algorithm, where every DSN performs FGA-FEC decode/recode. This method provides the best quality, since at each step the protection is optimal. In this section, we compare FGA-FEC adaptation vs. hop-by-hop FGA-FEC decode/recode in a congested multicast scenario. We use the ns-2 network topology of [53] shown in Fig 8.8, where the network backbone is congested (1.15 Mbps between source and node1). In this topology, we set the probability of packet drop of each link to

$p = 0.01$.

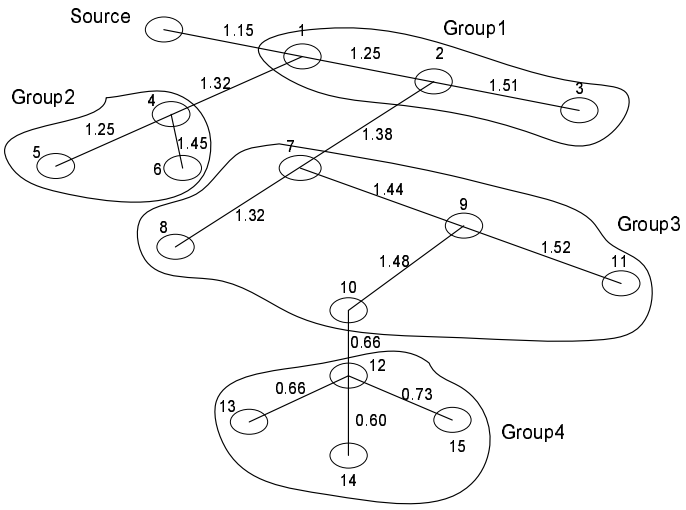
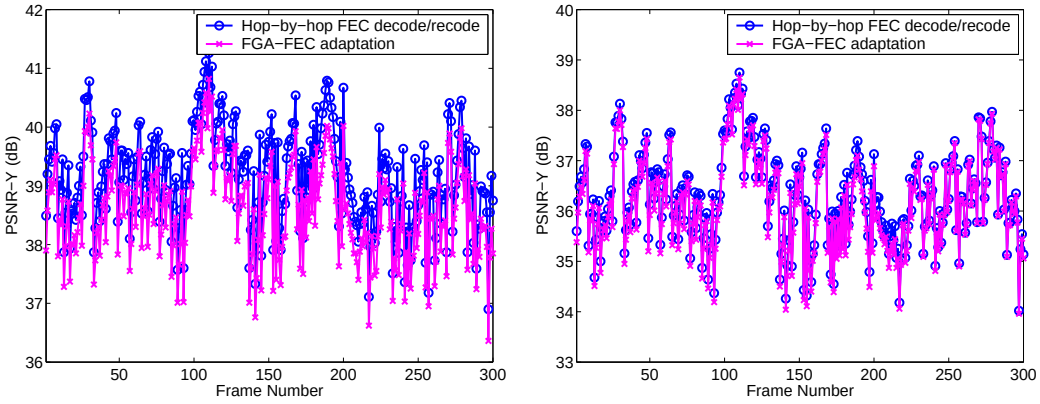


Figure 8.8: Network topology for a network of 16 nodes (link bandwidths are in Mbps, each link has a packet loss probability of 0.01). The backbone is congested, with smaller bandwidth than some end-user links.

In the hop-by-hop FGA-FEC decode/recode scheme, each DSN optimizes video for its down links and does FGA-FEC decode/recode for its direct children. In the FGA-FEC adaptation, the server uses the backbone available bandwidth 1.15 Mbps and aggregated packet-loss probability, roughly 0.07, to optimize the FGA-FEC encoding and then sends the encoded video into the network. All intermediate nodes perform possibly default FGA-FEC adaptation for their downlinks. Fig. 8.9 shows the PSNR quality delivered to receivers 5 and 12 respectively. In receiver 5, hop-by-hop FGA-FEC decode/recode performs better. The reason is that FGA-FEC needs to encode the scalable video at the combine network conditions, in this case, $B = 1.15$ Mbps and packet-loss probability approximately 0.07. On the other hand, the hop-by-hop FGA-FEC decode/recode scheme only needs to handle each individual link with its packet-loss probability 0.01. Regarding receiver 12, the PSNR loss is not as much as that at receiver 5, because the backbone is not congested compared with the user link's available bandwidth which is only at 0.66 Mbps. Here, the PSNR loss is mainly caused by the intermediate adaptation.

Results in this Section show that the hop-by-hop FGA-FEC decode/recode scheme outperforms FGA-FEC adaptation in a congested network, especially in the



(a) Receiver 5 (b) Receiver 12

Figure 8.9: PSNR quality delivered to two receivers. For receiver 5, FGA-FEC adaptation is about 0.56 dB lower than the distributed algorithm. For receiver 12, FGA-FEC adaptation is about 0.14 dB lower on average.

case of users with high quality requirements.

8.3.3 Comparison of Distributed FGA-FEC with Hop-by-hop FGA-FEC Decode/recode

The distributed FGA-FEC scheme uses OM-FEC to partition congested networks into segments and appropriately selects FGA-FEC decode/recode nodes while trying to maintain a near optimal delivered video quality. In this section, we compare hop-by-hop FGA-FEC decode/recode versus the distributed FGA-FEC scheme in a congested multicast scenario. We use the same ns-2 network topology in the previous section. For this topology, we set the probability of packet drop on each link to $p = 0.01$.

In the hop-by-hop FGA-FEC decode/recode, all intermediate nodes are involved in FEC decode /recode for their direct children. In the distributed FGA-FEC scheme, the congested network is partitioned into segments and only appropriate FGA-FEC decode/recode nodes are selected. In this topology, distributed FGA-FEC would identify three congested links in the backbone: (1) from source to node 1, (2) from node 1 to node 2, and (3) from node 10 to node 12. Thus, nodes 1, 2, 10, 11 are FGA-FEC decode/recode nodes. In the hop-by-hop FGA-FEC decode/recode method, all intermediate nodes are involved in FEC decode/recode, for

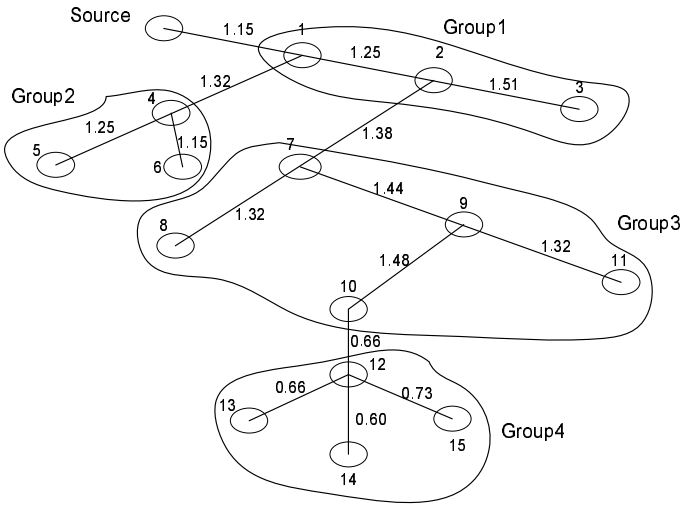


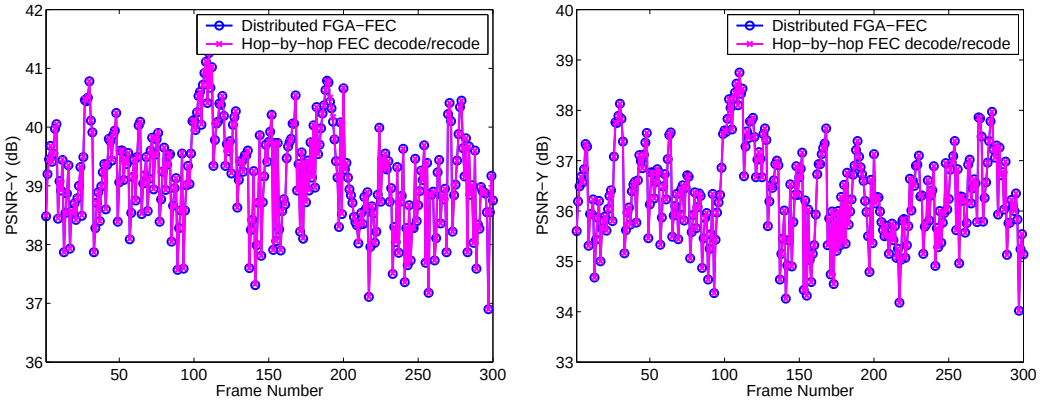
Figure 8.10: Network topology for a network of 16 nodes (link bandwidths are in Mbps, each link has packet-loss probability of 0.01). The backbone is congested, with smaller bandwidth than some end users.

a total of 7 nodes. Fig. 8.11 shows the PSNR quality delivered to receivers 5 and 12, respectively. For receiver 5, hop-by-hop FGA-FEC decode/recode is about 0.01 dB better on average than the distributed FGA-FEC, with the relative loss mainly caused by the better performance of FGA-FEC decode/recode at node 1. For hop-by-hop FGA-FEC decode/recode, the received video is FGA-FEC re-coded at node 4 with packet loss probability $p = 0.01$. For the distributed FGA-FEC, the video is FGA-FEC re-coded at node 1 with a aggregated loss probability of about $p = 0.02$. Regarding receiver 12, these two schemes perform about the same, since they both do FGA-FEC decode/recode at node 10 specifically for node 12.

The results in this Section show that the distributed FGA-FEC algorithm can provide similar quality to hop-by-hop FGA-FEC decode/recode, but with less than half the nodes involved in FEC computation, 7 nodes in FGA-FEC decode/recode versus 4 in distributed FGA-FEC in the section's simulation.

8.3.4 Distributed FGA-FEC CPU-Time

In the distributed FGA-FEC algorithm, a DSN either does FGA-FEC adaptation or FGA-FEC decode/recode to satisfy its users. Here, we compare the computational complexity of FGA-FEC adaptation vs. FGA-FEC decode/recode. We



(a) Receiver 5 (b) Receiver 12

Figure 8.11: Quality delivered in PSNR (dB) at two receivers. For receiver 5, distributed FGA-FEC average performance is less than 0.01 dB lower than the hop-by-hop FGA-FEC decode/recode algorithm. At receiver 12, both schemes have about the same video quality since they both do transcoding at parent node.

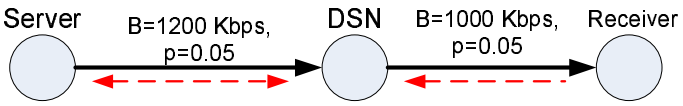


Figure 8.12: A simple topology video streaming to one user through DSN.

measured the CPU time of both schemes using the topology of Fig. 8.12, where the DSN is a Dell desktop with Pentium 4, 1.6 GHz CPU, 256 MB Memory, running Red Hat Linux 8.2. We use test sequences *Foreman* and *Mobile*, 7 GOPs/sequence, and number of descriptions encoded for each GOP N =64. The $D(R)$ curve rate interpolation interval is 100 bytes. The number of $D(R)$ points actually transmitted is 21.

FGA-FEC decode/recode	**FGA-FEC adaptation**
FGA-FEC decode time, FGA-FEC optimization time, and FGA-FEC recode time	The time to find the appropriate combination of dropping/shortening packets

Table 8.4: The measured items in FGA-FEC decode/recode and FGA-FEC adaptation methods

8.3.4.1 FGA-FEC Decode/recode Scheme

In the FGA-FEC decode/recode scheme, the server first does optimization over a channel (B = 1200 Kbps and p = 0.05), then encodes the video using RS codes. At the DSN, RS decoding is first performed, then it does the optimization for its downlink channel (B = 1000 Kbps and p = 0.05), finally it does the RS encoding.

FEC optimization time

To optimize FEC protection for each GOP, the DSN needs to: (1) interpolate the $D(R)$ curve, (2) convexify $D(R)$ curve and calculate related parameters such as α_i, q_i, (3) perform bisection search at appropriate initial λ value (one step of λ value calculation is called as one iteration), and (4) output the results. Table 8.5 shows the measured CPU time (in ms) of the different steps of the optimization algorithm for the two test clips.

Sequences	Total Optimization time/GOP (ms)	Bisection search time/GOP (ms)	Time/iteration (ms)
Foreman, FTFS	8.1	3.5	0.4
Foreman, HTFS	7.1	3.5	0.4
Foreman, FTHS	7.4	3.5	0.4
Mobile, FTFS	7.8	3.5	0.4

Table 8.5: Optimization CPU time. Here FTFS means full frame-rate full resolution, HTFS means half frame-rate full resolution and FTHS denotes full frame-rate half resolution. We show the average optimization time per GOP (sum of all four steps), the bisection search time, and the CPU time per iteration.

FGA-FEC decode/recode time

Table 8.6 shows the measured CPU time (in ms) of RS decode/recode at the intermediate node for *Foreman*, only first 7 GOPs.

8.3.4.2 FGA-FEC Adaptation Time

In the FGA-FEC adaptation, the server does the optimization and RS encoding based on aggregated network condition which is 1200 Kbps and p = 0.1. At the DSN, the FGA-FEC encoded bitstream is adapted for its downlink with 1000

Sequences	decode time (ms)	recode time (ms)	total (ms)
Foreman, FTFS	28.7	15.8	44.5

Table 8.6: Measured CPU time (in ms) of RS decode/recode at intermediate node. Results show that to perform FGA-FEC decode/recode takes 44.5 ms on average per GOP.

Kbps, $p = 0.05$ only by shortening packets and dropping descriptions. Here, we measure the CPU time to find the appropriate combination of dropping/shortening packets. In this scheme, the DSN needs to: (1) interpolate the $D(R)$ curve, (2) find the appropriate combination of dropping/shortening packets, and (3) output the results. Table 8.7 shows the measured CPU time of FGA-FEC adaptation.

Sequences	FGA-FEC adaptation time (ms)
Foreman, FTFS	2.9
Foreman, HTFS	1.8
Foreman, FTHS	1.9
Mobile, FTFS	2.6

Table 8.7: Measured CPU time (ms) of FGA-FEC adaptation

In summary, Table 8.8 compares the CPU time of running FGA-FEC decode/encode scheme and FGA-FEC adaptation on *Foreman* for the first 7 GOPs.

Scheme performed in DSN	CPU time (ms)
FGA-FEC decode/recode	52.6
FGA-FEC adaptation	2.9
FGA-FEC direct truncation	1×10^{-2}

Table 8.8: Intermediate node FGA-FEC decode/recode vs. FGA-FEC adaptation in terms of CPU time.

If the FGA-FEC direct truncation method (how many bytes need to be truncated from each packet) is used, the CPU time to process one GOP is less than 10^{-2}ms.

In our distributed FGA-FEC method, we coordinate between optimization processes running at adjacent nodes to reduce the number of iterations needed to

reach the stopping point. The DSNs usually need 2-3 iterations for each user in our distributed scheme vs. 8-10 iterations for a full search without coordination, thus we can save 30-40% CPU time in the optimization computation, or about 3 ms for each user. The optimization time saving is even more significant with FGA-FEC adaptation. The 3 ms saving is comparable to one FGA-FEC adaptation (2.9 ms) but much larger than direct truncation ($< 10^{-2}$ ms). In the FGA-FEC decode/recode case, FEC coding dominates the computation, about 52.6 ms. Since we only need to do one decoding and many encodings, for each encoding, the CPU time is about 16 ms, the total savings for one user is about $3 \div 16 = 20\%$. In addition to the coordination method, we apply the idea of OM-FEC to reduce the number of nodes involved in FEC decoding/recoding. The gain in the latter method is very significant, since the decoding/recoding time is ten of times longer than that of FGA-FEC adaptation (52 ms vs. 2.9 ms). i.e. each DSN can support ten times as many users if using FGA-FEC adaptation than using FGA-FEC decode/recode.

8.4 Conclusion

In this chapter, we proposed a distributed FGA-FEC algorithm for video streaming to diverse users on a congested network. We proposed a distributed approach to greatly reduce the computational burden of optimization by exchanging overhead information between adjacent nodes. We also extended the idea of OM-FEC to determine the congested links and hence to reduce the number of needed FGA-FEC decode/encode nodes. Here we apply FGA-FEC adaptation whenever permitted and do FGA-FEC decode/recode only at the edge of congested links. Simulations have shown the performance of the proposed scheme.

CHAPTER 9
Contributions and Suggested Future Work

9.1 Contributions

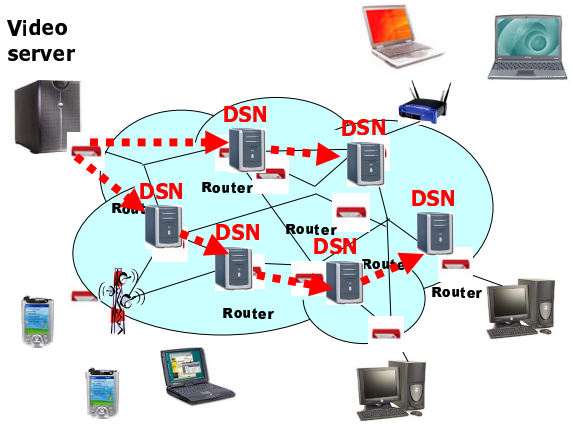


Figure 9.1: Streaming video to heterogeneous users through an overlay network, where DSNs are data service nodes with certain functionalities

We investigated several techniques to efficiently support video streaming to heterogeneous users over both wired and wireless networks, as shown in Fig. 9.1, where DSNs are data service nodes with certain functionalities. The main contributions of this thesis are described in detail as follows.

9.1.1 Fine Grain Adaptive Forward Error Correction

In this study [98, 99], we investigate a fine grain adaptive forward error correction (FGA-FEC) coding scheme for both channel coding and adaptation of scalable video bitstreams. In our work, both the embedded source bitstream and the error-control codes are easily and precisely adapted at intermediate overlay nodes to satisfy multiple heterogeneous users without complex transcoding. The proposed FGA-FEC scheme encodes and adapts the scalable bitstream in such a way that if part of the video source data is actively dropped, parity bits protecting that piece of data are also removed, yielding an efficient result without FEC transcoding. Encoding once at the source, the new method can satisfy multiple heterogeneous users

simultaneously without decoding/recoding FEC at intermediate network nodes.

9.1.2 Generalized FGA-FEC over Wireless Networks

In this study [103], we extend our proposed FGA-FEC coding scheme, a generalized MD-FEC method, to wireless networks. To protect the encoded scalable video bitstream over lossy channel and facilitate content adaptation at intermediate nodes, we use product codes based on BCH/CRC codes as row codes and RS codes as column codes. We propose a fast algorithm to optimize the product codes within several iterations from a near optimal point. Simulations show good performance in both content adaptation and protection.

9.1.3 Improving Multimedia Throughput using Header Error Protection in WLANs

In the generalized FGA-FEC, since we assume that packet with errors are passed to application layer for error correction, we need to have a method to support this. In this study [105, 104], we propose two link layer error protection schemes (header CRC and header FEC) that improve the effective throughout of wireless networks. Error control is applied to the packet header (at link layer) and packet payload (at application layer) separately. The network intermediate nodes either use header FEC or header CRC checksum to successfully transport packets from the source to the destination. We compare the proposed schemes with conventional IEEE 802.11 protocol which is designed for reliable data communication. Both theoretical analysis and ns-2 simulation results show that header error protection strategy can effectively increase the application throughput.

9.1.4 Cross-layer Two-stage FEC Scheme

In this study [102], we propose a cross-layer two-stage FEC scheme in cooperation with the enhanced MAC protocol (header CRC/FEC) especially for multimedia data transmission over wireless LANs. The proposed scheme enables the joint optimization of protection strategies across the protocol stack. In stage 1, packet-level FEC is added across packets at the application layer to correct packet losses due to congestion and route disruption. In stage 2, bit-level FEC is processed within both

application packets and stage-one FEC packets to recover from bit errors in the MAC/PHY layer. Header CRC/FEC schemes are used to enhance the MAC/PHY layer and to cooperate with the two-stage FEC scheme. Thus, we add FEC only at the application layer, but can correct both application layer packet drops and MAC/PHY layer bit errors. We explore both the efficiency of bandwidth utilization and video performance using the scalable video coder MC-EZBC and ns-2 simulations. Simulation results show the efficiency of the proposed scheme.

9.1.5 Overlay Multi-hop FEC Scheme

In both FGA-FEC and its wireless extension schemes, we did not consider the case of network congestion in backbone. In this study [100, 101], we focus on the problem of providing lightweight support at selected intermediate overlay forwarding nodes to achieve increased error resilience on a single overlay path for video streaming. We propose a novel overlay multi-hop forward error correction (OM-FEC) scheme that provides FEC encoding/decoding capabilities at some intermediate nodes in the overlay path. Based on the network conditions, the end-to-end overlay path is partitioned into segments, and appropriate FEC codes are applied over those segments. Architecturally, this flexible design lies between the end-to-end and hop-by-hop paradigms, and we argue that it is well suited to peer-based overlay networks. We evaluate our work by both simulations and controlled PlanetLab network experiments. These evaluations show that OM-FEC can outperform a pure end-to-end strategy up to 10-15 dB in terms of video peak signal-to-noise ratio (PSNR), and can be much more efficient than a heavyweight hop-by-hop strategy, in which all the overlay nodes along the path are involved in FEC computation.

9.1.6 Distributed FGA-FEC Scheme

In the previous chapter, we proposed an OM-FEC method to efficiently utilize one congested overlay path. In this chapter, we investigate a distributed FGA-FEC scheme over a congested multihop network, where we do FGA-FEC decode/recode at selected intermediate overlay nodes, and do FGA-FEC adaptation at remaining nodes. In order to reduce the overall computational burden, we proposed two methods: (1) a coordination between optimization processes running at adjacent nodes

to reduce the optimization computation, and (2) apply OM-FEC from Chapter 7 to reduce the number of FGA-FEC decode/recode nodes. Simulations show that the proposed scheme can greatly reduce computation, and can provide near best possible video quality to users.

9.2 Suggested Future Work

Having summarized the contributions of this thesis, we outline several possible directions for future research.

(1) We propose FGA-FEC scheme to work with a highly scalable subband/wavelet based video coder over heterogeneous networks. The encoded embedded bitstream has a very fine grained SNR scalability, so that we can manipulate the bitstream to fit all kind of network conditions. We did not evaluate the FGA-FEC idea over coarse DCT-based layered video bitstream, such as SVC with only several SNR layers. We need to evaluate FGA-FEC idea with current standard DCT-based video coders, such as H.264/AVC and SVC etc.

(2) In Chapter 8, we propose a distributed FGA-FEC scheme over congested multihop network, where we only consider SNR scalability. Future work may focus on multi-dimensional scalability, i.e. users have not only video request about SNR scalability, but also video request with a certain temporal and spatial constraint.

(3) The proposed FGA-FEC scheme mainly focus on streaming video to heterogeneous users. Can it be extended to work with multi-points video conferencing? Multi-point video conferencing is a real-time application, delay larger than 300ms usually is not acceptable. We may modify FGA-FEC scheme to work with real-time applications by encoding a smaller number of pictures, instead of a large GOP. This is also a possible extension.

LITERATURE CITED

[1] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS*, pp. 1-12, Montery, CA, 2000.

[2] B. Raman, S. Agarwal, and et al, "The SAHARA model for service composition across multiple providers," in *Proc. of International Conference on Pervasive Computing*, 2002.

[3] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "CANS: Composable, adaptive network services infrastructure," in *Proc. of USENIX Symposium on Internet Technologies and Systems*, 2001.

[4] E. A. Riskin, "Optimal bit allocation via the generalized BFOS algorithm," *IEEE Trans. on Information Theory*, vol. 37, pp. 400-402, March 1991.

[5] I. V. Bajic, Robust subband/wavelet coding and transmission of images and video, Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY, 2003.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," ICSI, Tech. Rep. TR-00-03, March 2000.

[7] X. Gu and K. Nahrstedt, "Qos-assured service composition in managed service overlay networks," in *Proc. of International Conference on Distributed Computing Systems*, pp. 194-201, Providence RI, May 2003.

[8] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Distributing streaming media content using cooperative networking," Microsoft Corporation, WA, Tech. Rep. MSR-TR-02-37, 2002.

[9] The MPEG-21 website. [Online]. Available: http://www.chiariglione.org/mpeg/

[10] A. Vetro and C. Timmerer, "Digital item adaptation: overview of standardization and research activities," *IEEE Trans. Multimedia (special MPEG-21 issue)*, April 2005.

[11] ISO/IEC 15444-1, "JPEG2000 image coding system," 2000.

[12] ISO/IEC 11172-2, "Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 2: Video," 1993

[13] ISO/IEC 13818-2, "Information technology - Generic coding of moving pictures and associated audio information: Video," 2000.

[14] ISO/IEC 14496-2, "Information technology - Coding of audio-visual objects - Part 2: Visual," 1999.

[15] International Telecommunication Union, ITU-T Recommendation H.261, "Video codec for audiovisual services at px64 kbits," 1993.

[16] International Telecommunication Union, ITU-T Recommendation H.263, "Video codeing for low bitratecommunications version 2," 1998.

[17] International Telecommunication Union, ITU-T Recommendation H.264, "Advanced Video Coding for Generic Audiovisual Services", 2003.

[18] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: John Wiley & Sons, 1991.

[19] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic coding for data compression," *ACM Communications*, vol. 30,pp. 520-540, June 1987.

[20] G.D. Karlsson, M. Vetterli, "Three dimensional subband coding of video", in *Proceedings of ICASSP*, pp. 1100-1103, 1988

[21] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Processing*,vol. 3, pp. 559-571, Sept. 1994

[22] S. Choi and J. W. Woods, "Motion-Compensated 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 8, no. 2, pp. 155-167, Feb. 1999

[23] S.T Hsiang and J.W. Woods, "Invertible three-dimensional analysis/systhesis system for video coding with half-pixel-accurate motion compensation", in *Proceedings of SPIE VCIP*, Vol. 3652, pp. 537-546, San Jose, CA, 1999

[24] B. Pesquet-Popescu and V. Bottreau, "Three-dimensional lifting schemes for motion compensated video compression," in *Proceedings of ICASSP*, pp. 1793-1796, May 2001.

[25] A. Secker and D. Taubman, "Motion-compensated Highly Scalable Video Compression Using An Adaptive 3-D Wavelet Transform Based on Lifting," in *Proceedings of ICIP*, October 2001

[26] J. W. Woods and P. Chen, "Improved MC-EZBC with Quarter-pixel Motion Vectors," ISO/IEC JTC1/SC29/WG11, MPEG2002/8366, Fairfax, USA, May 2002.

[27] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 8, pp. 1374-1387, December 2000.

[28] S.-T. Hsiang and J. W. Woods, "Embedded video coding using invertible motion compensated 3-D subband/wavelet filter bank," *Signal Processing: Image Commun.*, vol. 16, no. 8, pp. 705-724, May 2001.

[29] S.-T. Hsiang, *Highly scalable subband/wavelet image and video coding*, Ph.D Thesis, Rensselaer Polytechnic Institute, Troy, NY, January 2002.

[30] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing* ,vol. 1, pp. 205-220, April 1992.

[31] D.Taubman and A.Zakhor, "Multirate 3-D Subband Coding of Video," *IEEE Trans. Image Processing*, vol.3, pp.57-88, September 1994.

[32] J. M. Shapiro, "An embedded hierarchical image coder using zerotrees of wavelet coefficients," in *Proc. IEEE Data Compression Conference*, pp. 214-223, Snowbird, Utah, USA, MarchC April 1993.

[33] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243-250, 1996.

[34] W. Li, "Overview of fine granular scalability inMPEG-4 video standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301-317, 2001.

[35] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video processing and communications*, Prentice-Hall, 2002.

[36] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Transactions on Information Theory*, vol. 42, pp. 1737-1744, Nov. 1996.

[37] A. E. Mohr, R. E. Ladner, and E. A. Riskin, "Approximately optimal assignment for unequal loss protection," in *Proc. ICIP*, Vancouver, BC, September 2000

[38] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," in *Proc. ICIP*, Rochester, NY, September 2002.

[39] R. Puri, K. Ramchandran, "Multiple description coding using forward error correction codes", in P*roc. 33rd Asilomar Conf. on Signals and Systems*, Pacific Grove, CA, Oct. 1999.

[40] T. Stockhammer, C. Buchner, "Progressive texture video streaming for lossy packet networks," in *Proc. 11th International Packet Video Workshop*, Kyongju, May 2001.

[41] S. Dumitrescu, X. Wu, Z. Wang, "Globally optimal uneven error-protected packetization of scalable code streams", in *Proc. DCC02*, Snowbird, Utah, April 2002.

[42] D. Mukherjee, A. Said, and S. Liu, "A framework for fully formatindependent adaptation of scalable bit streams," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1280-1290, Oct. 2005.

[43] D. Mukherjee, E. Delfosse, J. Kim, and Y. Wang, "Optimal adaptation decision-taking for terminal and network quality of service," (Invited paper) *IEEE Trans. on Multimedia*, pp. 454-462, June. 2005.

[44] The HP SSM website. [Online]. http://www.hpl.hp.com/research/ssm

[45] B. Raman, S. Agarwal, *et al*, "The SAHARA model for service composition across multiple providers," in *Proc. of International Conference on Pervasive Computing.*

[46] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "CANS: Composable, adaptive network services infrastructure," in *Proc. of USENIX Symposium on Internet Technologies and Systems*, 2001.

[47] X. Gu and K. Nahrstedt, "Qos-assured service composition in managed service overlay networks," in *Proc. of International Conference on Distributed Computing Systems*, pp. 194-201, Providence RI, May 2003

[48] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologicallyaware overlay construction and server selection," in *Proc. INFOCOMM*, New York, NY, 2002.

[49] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329-350, Heidelberg, Germany, 2001,

[50] D. Xu and K. Nahrstedt, "Finding service paths in a media service proxy network," in *Proc. of SPIE/ACM Multimedia Computing and Networking Conference*, San Jose, CA, 2002.

[51] P. Chen, S. Hsiang, J. Woods, D. Mukherjee, G. Kuo, and A. Said, "Fully scalable mc-ezbc in the structured scalable meta-formats (ssm) framework," in ISO/IEC JTC1/SC29/WG11 MPEG2002/M9290, Awaji, Japan, Dec. 2002.

[52] P. Puri, K. W. Lee, K. Ramchandran, and V. Bharghavan, "An integrated source transcoding and congestion control paradigm for video streaming in the interent," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 18-32, March 2001.

[53] R. Puri, K. Ramchandran, K.W. Lee and V. Bharghavan, "Forward error correction codes based multiple description coding for internent video streaming and multicast," *Signal Processing: Image Commincations*, vol 16, pp. 645-276, 2001.

[54] S.Ratnasamy, S. McCanne, "Scaling end-to-end multicast transports with a topologically-sensitive group formation protocol" ICNP, Toronto, Oct. 1999.

[55] I. V. Bajic and J. W. Woods, "EZBC video streaming with channel coding and error concealment," in *Proc. SPIE VCIP*, July 2003, pp. 512-522.

[56] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", *IEEE Trans. on Information Theory*, IT-46, pp. 1204-1216, 2000.

[57] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding". *IEEE Transactions on Information Theory*, Februray, 2003

[58] T. Ho, M. Medard, M. Effros, J. Shi, and R. Koetter, "Toward a random operation of networks," *IEEE Transactions on Information Theory*, 2004.

[59] T. Ho, R. Koetter, M. Mdard, D. R. Karger and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," *IEEE International Symposium on Information Theory*, 2003.

[60] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference on Communication, Control, and Computing*, Monticello,IL, 2003

[61] P.A. Chou, A. E. Mohr, A. Wang, S. Mehrotra, "FEC and pseudo-ARQ for receiver-driven layered multcast of audio and video," in *Proceedings of the DCC*, Snowbird, UT, March 2000

[62] S.R. McCanne, "Scalable compression and transmission of internet multicast video," Ph.D thesis, University of California, Berkeley, Berkeley, CA 1996.

[63] C. Gkantsidis, P. Rodriguez, "Network Coding for Large Scale Content Distribution", in *Proc. INFOCOM* 2005, Miami. March 2005. (Also as Microsoft Research Technical Report, MSR-TR-2004-80)

[64] http://www.bittorrent.com/

[65] K. Yeo, B.S. Lee, M.H. Er, "A Peering architecture for ubiquitous IP multicast streaming," in *ACM SIGOPS Oper. Systems Rev.* 36 pp. 82-95, (July 2002).

[66] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001, pp. 55-67.

[67] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Morris, "Resilient overlay networks," in *Proceedings of the ACM SOSP*, pp. 131-145, Banff, Canada, October 2001

[68] U. Horn, K. Stuhlmuller, M. Link, B. Girod, "Robust internet video transmission based on scalable coding and unequal error protection," *Signal Processing: Image Commun.* pp. 77-94, September 1999

[69] W. Tan, A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Trans. Circuits Systems Video Technol.* pp. 373-386, March 2001.

[70] T. Nguyen, A. Zakhor, "Distributed video streaming with forward error correction," *Packet Video Workshop*, Pittsburgh, PA, April 2002.

[71] T.Nguyen, A. Zakhor, "Distributed video streaming over the internet," in *Proceedings of the SPIEConference on Multimedia Computing and Networking*, San Jose, California, January 2002.

[72] J. Kim, R.M. Mersereau, Y. Altunbasak, "Distributed video streaming using unbalanced multiple description coding and forward error correction," *IEEE Globecom*, San Francisco, CA, Dec. 2003.

[73] J. Apostolopoulos, W. Tan, S. Wee, G. Wornell, "Modeling path diversity for multiple description video communication, in *Proceedings of the IEEE International Conference on Acoustics Speech Signal Processing*, May 2002.

[74] A.C. Begen, Y. Altunbasak, O. Ergun, M.H. Ammar, "Multi-path selection for multiple description video streaming over overlay networks," *Signal Process.: Image Commun.* 20/1, pp. 39-60, January 2005 .

[75] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proceeding of the ACM SIGCOMM*, pp. 149-160, San Diego, CA, August 2001, .

[76] J.M. Boyce, R.D. Gaglianello, "Packet Loss Effects on MPEG Video Sent over the Public Internet," in *Proc. ACM Multimedia*, pp. 181-190, Bristol, UK, September 1998.

[77] V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Trans. Network.* pp. 277C292, June 1999

[78] W. Tan, A. Zakhor, "Multicast transmission of scalable video using receiver-driven hierarchical FEC," in *Proc. Packet Video Workshop*, April 1999.

[79] S.B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, Englewood cliffs, NJ, 1995.

[80] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, B. Basch, "Robust compression and transmission of MPEG-4 video," in *Proceeding of ACM Multimedia*, pp. 113-120, Orlando, Florida, June 1999, .

[81] P. Karn, General-purpose ReedCSolomon encoder/decoder in C, version 4.0, available at http://www.ka9q.net/code/fec/.

[82] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Comput. Commun. Rev.* vol. 27, pp. 24-36, April 1997

[83] http://www.planet-lab.org/

[84] J. Bolot, S. Fosse-Parisis, D. Towsley, "Adaptive FEC-based error control for internet telephony," in *Proceedings of the IEEE INFOCOM*, pp. 1453-1460, New York, March 1999,

[85] K. Stuhlmuller, N. Farber, M. Link, B. Girod, "Analysis of video transmission over lossy channels," *IEEE J. Select Areas Commun.* vol. 18 pp. 1012-1032, June 2000

[86] J. Padhye, V. Firoiu, D. Towsley, J. Krusoe, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of the ACM SIGCOMM*, pp.303-314, Vancouver, September 1998

[87] IEEE Computer Society LAN MAN Standard Committee, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std. 802.11-1999, New York, 1999.

[88] Q. Li and M. van der Schaar, "Providing adaptive qos to layered video over wireless local area networks through real-time retry limit adapation," *IEEE Transactions on Multimedia* vol. 6, pp. 278-290, April 2004.

[89] S. Krishnamachari, M. van der Schaar, S. Choi, and X. Xu, "Video streaming over wireless LANs: a crosslayer approach," in *Proc. Packet Video*, April 2003.

[90] M. H. Manshaei, T. Turletti, and T. Guionnet, "An evaluation of media-oriented rate selection algorithm for multimedia transmission in MANETs," *EURASIP Journal on Wireless Communications and Networking*, March 2005.

[91] A. Goldsmith, E. Setton, T. Yoo, X. Zhu, and B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," *IEEE Wireless Communications Magazine*, invited paper, 2005.

[92] L. Choi, W. Kellerer, and E. Steinbach, "Cross layer optimiztion for wireless multi-user video streaming," in *Proceeding of International Conference on Image Processing (ICIP2004)*, December 2004.

[93] "RFC 3828: The lightweight user datagram protocol (udp-lite)," July 2004.

[94] IETF Manet Working Group AODV Draft: http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt

[95] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.* 39, pp. 1253-1265, September 1960.

[96] The network simulator- ns-2; http://www.isi.edu/nsnam/ns/.

[97] Video clips; http://networks.ecse.rpi.edu/ yfshan/videoITcom05/.

[98] Yufeng Shan, Ivan Bajic, Shivkumar Kalyanaraman, and John W. Woods, "Joint Source-Network Error Control Coding for Scalable Overlay Streaming," in *proceedings of ICIP* 2005

[99] Yufeng Shan, Ivan Bajic, Shivkumar Kalyanaraman, and John W. Woods, "Scalable Video Streaming with Fine Grain Adaptive Forward Error Correction," submitted to *IEEE Trans. CSVT* 2006

[100] Yufeng Shan, Ivan Bajic, Shivkumar Kalyanaraman, and John W. Woods, "Overlay Multi-hop FEC Scheme for Video Streaming over Peer-to-Peer Networks," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Vol. 5, Pages 3133-3136, Singapore, October 2004

[101] Yufeng Shan, Ivan V. Bajic, Shivkumar Kalyanaraman and John W. Woods, "Overlay Multi-hop FEC scheme for Video Streaming," *Elsevier Journal on Signal Processing: Image Communications, Special Issue on Video Networking,* Vol 20/8, pp. 710-727, 2005

[102] Yufeng Shan, Su Yi, Shivkumar Kalyanaraman and John.W. Woods, "Two-Stage FEC Scheme for Scalable Video Transmission over Wireless Networks" *SPIE Communications/ ITCom, Multimedia Systems and Applications.* Oct. 2005, Boston, MA

[103] Yufeng Shan, John W. Woods and Shivkumar Kalyanaraman, "Fine grain adaptive FEC (FGA-FEC) over wireless networks", submitted to ICIP 2007

[104] Su Yi, Yufeng Shan, Shivkumar Kalyanaraman and Babak Azimi-Sadjadi, "Header Error Protection for Multimedia Data Transmission in Wireless Ad Hoc Networks", in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Atlanta GA, Oct. 2006

[105] Su Yi, Yufeng Shan, Shivkumar Kalyanaraman and Babak Azimi-Sadjadi, "Video Streaming over 802.11 Ad Hoc Wireless Networks with Header Error Protection", submitted to *Ad Hoc Networks*, Dec. 2005

[106] D. Mukherjee, A. Said, and S. Liu, "A framework for fully formatindependent adaptation of scalable bit streams," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1280-1290, Oct. 2005.

[107] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Robust layered multiple description coding of scalable media data for multicast," *IEEE Signal Processing Letters*, vol. 12, no. 2, pp. 154-157, Feb. 2005.

[108] J. W. Woods, P. Chen, Y. Wu, and S.-T. Hsiang, *Interframe Subband/ Wavelet Scalable Video Coding*, in Handbook of Image and Video Processing. Burlington, MA: Elsevier Academic Press, 2005.

[109] H. Zheng and J. Boyce, "An improved UDP protocol for video transmission over internet-to-wireless networks," *IEEE Trans. on Multimedia*, vol. 3, no. 3, pp. 356-365, September 2001.

[110] D. A. Eckhardt and P. Steenkiste, "Improving wireless LAN performance via adaptive local error control," in *Proc. ICNP*, 1998.

[111] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. on Information Theory*, vol. IT-46, no. 2, pp. 388-404, March 2000.

[112] E.O.Elliot, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, pp. 1977-1997, September 1963.

[113] M. Zorzi, R. R. Rao, and L. B. Milstein, "On the accuracy of a first-order markov model for data transmission on fading channels," *in Proceedings of ICUPC*, pp. 211-215, Tokyo, Japan, Nov. 1995

[114] Shu Lin and Daniel J. Costello, *Error Control Coding*, 2nd Ed., Pearson Education, 2004.

[115] J. Li, C. Blake, D. D. Couto, H. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proc. ACM MobiCom*, July 2001.

[116] S. Yi, *Error Control Schemes and Directional Antennas in Wireless Networks*, Ph.D thesis, Department of ECSE, Rensselaer Polytechinic Institute, Dec. 2005

[117] P.G. Sherwood and K. Zeger, "Error protection for progressive image transmission over memoryless and fading channels," *IEEE Trans. Comm.* vol. 46, pp. 1555-1559, Dec. 1998

[118] V. Stankovic, R. Hamzaoui, Z. Xiong, "Product code error protection of packetized multimedia bitstreams", in *Proc. IEEE International Conference on Image Processing*, Barcelona, September 2003.

[119] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Real-Time Error Protection of Embedded Codes for Packet Erasure and Fading Channels", *IEEE Transaction on circuits and systems for video technology*, vol. 14, no. 8, august 2004

[120] S. Cho and W. A. Pearlman, "Multilayered Protection of Embedded Video Bitstreams over Binary Symmetric and Packet Erasure Channels," *J. Visual Communication and Image Representation*, Vol. 16, pp. 359-378, June 2005

[121] P. Greg Sherwood and Kenneth Zeger, "Error Protection for Progressive Image Transmission Over Memoryless and Fading Channels", *IEEE Transections on commumication*, vol. 46, no. 12, Dec. 1998

[122] D. G. Sachs, R. Anand, K. Ramchandran, "Wireless Image Transmission Using Multiple-Description Based Concatenated Codes", in *Proc. VCIP*, vol. 3974, pp. 300-311, San Jose, CA, Jan. 2000.

[123] R. Hamzaoui, V. Stankovic, and Z. Xiong, "Optimized Error Protection of Scalable Image Bit Streams" *IEEE Signal processing magazine*, pp. 91-107, Nov. 2005