JOINT SOURCE-CHANNEL VIDEO TRANSMISSION

BY

LEIMING QIAN

B.E., Tsinghua University, Beijing, 1996
M.S., University of Illinois at Urbana-Champaign, 1999

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2001

Urbana, Illinois

ABSTRACT

With the rapid growth of multimedia content in today's communication links, there is an increasing demand for efficient image and video transmission systems. Various research work has shown that significant gain in transmission performance can be achieved from the application of *joint source-channel matching* (JSCM) techniques, which maximize the final end-to-end performance by optimally allocating limited system resources (transmission power, channel capacity, etc.) between the source and channel coders.

This thesis investigates the application of JSCM to video transmission associated with the following three different types of communication channels:

1. **Peer transmission:** an adaptive matching technique, based on our earlier work of general source-channel matching techniques using parametric modeling, is presented for the case of direct point-to-point video transmission with a reliable feedback channel.

2. **Broadcasting:** based on the JSCM principle developed for the previous case of peer transmission, this thesis proposes an alternative performance criterion under the name of *minimax disappointment* (MD) for a multireceiver broadcasting scenario, where feedback channels do not exist.

3. **Network transmission:** assuming a network consisting of multiple video senders and receivers, this thesis presents a *joint source-network matching* (JSNM) transport layer protocol for video transmission. The JSNM protocol is complete with a custom packet format, a congestion-control algorithm which is *Transmission Control Protocol* (TCP) friendly, and built-in loss-and-error control mechanisms for both packet loss due to congestion and packet corruption due to wireless fading.

The effectiveness of the proposed JSCM techniques, MD optimization criterion and JSNM transport protocol is illustrated using simulations based on various video source coders and standard channel coders.

DEDICATION

To my parents, whose unconditional love has supported me through all these years of hardship, and to Qi, who brings so much joy into my otherwise dull life. This thesis is for all of you.

ACKNOWLEDGMENTS

This work has largely been supported by the U.S. National Science Foundation through grant no. MIP-9707742 and the Motorola Corporation.

I would like to express my immense gratitude toward my adviser, Professor Douglas L. Jones, for all the kindness, patience and understanding he has shown toward me, and all the helpful and insightful suggestions he has given me, both in research and in my development as a person. He is always there to listen, to advise, and to help. Working with him has made research both a fruitful and enjoyable experience.

I would like to thank the authors of the network simulation software package NS-2, for their excellent work and generosity in donating NS-2 to network research community.

My gratitude also goes to Henry Kwok and Mark Haun, for installing and maintaining our group's Linux workstation, on which all my network simulations have been carried out.

TABLE OF CONTENTS

CHAPTER **PAGE**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the recent advancement in signal processing and communication theory, also fueled by the rapid expansion in available channel bandwidth, multimedia communication (peer-to-peer, broadcasting, etc.), especially video communication (videophone, videoconference), is attracting increasing attention from both the academic and industrial world. It is desirable to discover a general method to design an effective video communication system.

It is widely accepted that Shannon's source-channel coding theorem [1], which states conditions under which the source and channel coders in a communication system can be optimized separately without sacrificing the system's overall performance, is the foundation of designing such a system. To that effect, the source coders are traditionally designed to achieve the maximum compression ratio (recently, more efforts have been seen in designing channel-robust coders), and the channel coders are designed to yield the smallest error probability given the transmission rate. There is no apparent need for information exchange or joint optimization between source and channel coders.

However, this theorem applies under the assumption of infinite-length codewords (implying infinite transmission delay) and point-to-point transmission only (no multi-path commonly observed in wireless communication), which is often not realistic in practical situations. These assumptions are especially invalid for video transmission, where by nature only a small amount of delay can be tolerated and the receiver has a finite-size decoding buffer. In practical video communication applications, where system resources are limited and constraints are posed on codeword length and tolerable transmission delay, jointly optimizing the source and channel coders can often bring significant end-to-end performance increase. This topic is explored in much greater depth in Section 2.1.

The Internet has been experiencing explosive growth of multimedia content, with most current applications involving web-based audio and video playback. This continuing trend of growth and the Internet's mounting popularity is a clear indication that the human society will

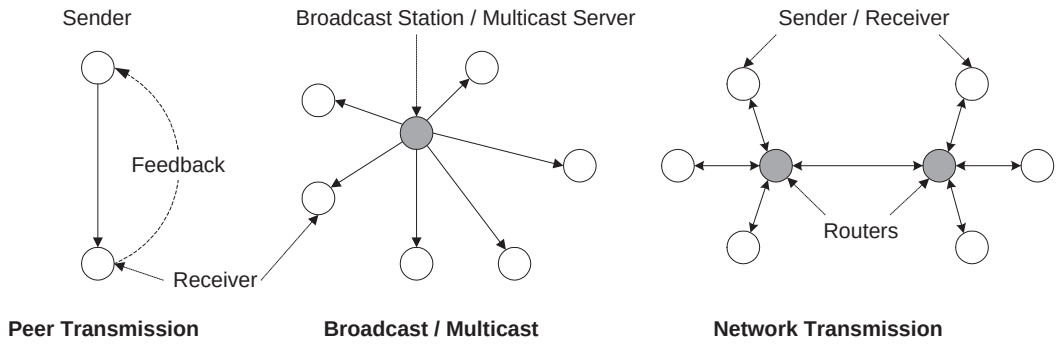

Figure 1.1 The joint source-channel matching research framework.

eventually become interconnected in an extraordinary fashion beyond what can be conveyed by simple text messages or even still images. In the future world of fast communication and interactive information exchange, animated visual communication over networks will almost surely play a major role. The basic principle of source-channel joint optimization can also be applied to video communication systems based on network channels [2], since network links are fundamentally special cases of the entire class of communication channels.

The design of an efficient joint source-channel (network) optimized system requires that one traverses three basic research areas: *video source coding*, *channel coding*, and *network systems* [3]. Therefore, a significant amount of detailed background knowledge needs to be covered. Technical issues crucial to the understanding of this thesis are addressed in Chapter 2: in Section 2.2 we categorize and list various video source coders, discuss their individual advantages and shortcomings; in Section 2.3 we address the important video-related issue of *rate control* and its impact on the implementation of JSCM; in Section 2.4, we briefly mention the basic characteristics of two widely-used channel codes, Reed-Solomon (RS) codes and *rate-compatible punctured convolutional codes* (RCPC); in Section 2.5, we give a short introduction to various network protocols. Other relevant detailed technical information can be found in the appendices.

Based on the type of channel and user configuration under investigation, we roughly divide our research activities into three major categories (also illustrated in Figure 1.1), with a focus on the last category:

1. **Peer transmission**: Simple direct point-to-point transmission consisting one sender and one receiver, with a *reliable feedback channel*. The existence of the feedback channel

and the guaranteed accuracy of the feedback information are crucial to the successful implementation of any joint source-channel matching (JSCM) peer transmission scheme.

2. **Broadcasting/multicasting**: One central broadcast service provider (or multicast server) transmits the same set of signals to multiple receivers (or clients) under different channel situations, no feedback channels available.

3. **Network transmission**: Concurrent bidirectional video transmission between multiple sender/receiver pairs over hybrid wireless/wireline network links; feedback channels may exist in either reliable form or unreliable form.

These three categories are not by any means independent or mutually exclusive; they are organized in this fashion merely for the convenience of presentation.

The following chapters are organized as follows: in Chapter 3 we develop and demonstrate the effectiveness of the basic JSCM system for simple peer transmission, which also serves as the foundation for the two following cases; in Chapter 4, based on the JSCM principle, we propose an alternative broadcast/multicast performance criterion named "minimax disappointment"; in Chapter 5 we discuss the network extension of the JSCM system, joint source-network matching (JSNM) system, based on the transport layer *Joint Source-Network Matching Protocol* (JSNMP) detailed in Chapter 6; we conclude in Chapter 7 .

Simulations results are presented and discussed at the end of each chapter associated with the specific type of communication channel it addresses. Related mathematical derivations and system implementation details are included in the appendices: Appendix A discusses the rate-distortion estimation procedure for various video source coders; Appendix B gives details on specific joint source-channel matching system implementations and optimization algorithms; Appendix C addresses our wireless video broadcasting system; Appendix D provides necessary theoretical knowledge about minimax optimization; and Appendix E describes our joint source-network matching transport protocol with implementation-level details.

1.1 Summary of Results and Contributions

In our research toward the topic of JSCM video transmission, we have reached the following results.

1. We have shown that JSCM can be successfully applied to point-to-point peer video transmission systems based on a wide variety of source-channel coder pairs. Significant end-to-

end performance gain can be achieved. A fast adaptive system implementation (Section 3.3) is developed as an extension of our earlier work on a general matching system (Section 3.2).

2. We have proposed an alternative JSCM-based broadcasting performance evaluation criterion: minimax disappointment (Section 4.2), which possesses several appealing properties such as being universally fair, able to utilize the concept of users' individual expectation.

3. We have designed a joint source-network matching video transmission system, based on a simple hybrid wireless / wireline network topology, a new transport protocol on top of the *User Datagram Protocol* (UDP), a TCP-friendly congestion control algorithm, and built-in packet loss-error control mechanisms.

1.2 Future Research Plans

As a typical cross-disciplinary research topic, there are endless exciting research possibilities for joint source-channel (network) video communication. To cover all these topics in this thesis would be beyond our capabilities. Here we list only a few which we consider might be of major interest to fellow JSCM researchers.

1. The practical application of joint source-network video transmission to different network setups: i.e, various existing network transport protocols, routing algorithms, etc. We seek a general design methodology for network video communication systems.

2. Network multicasting, the counterpart of wireless broadcasting, is another area to which we expect to successfully apply joint source-network matching (JSNM) and our minimax disappointment performance criterion.

3. In addition, the investigation of jointly optimizing the source and network can provide potentially beneficial insights to the design of source-network coders as well. Examples include network-robust video source coders, JSCM-aware network transport protocols and routing algorithms which can take JSCM information as hints, etc.

1.3 Impact of the Problem

We believe it is worthwhile, at the end of this introductory chapter, to briefly discuss the impact of this problem, and how its solution, once found, could be of great benefit.

Based on a simple and elegant concept, JSCM can truly provide meaningful performance gains under various situations when Shannon's information separation theorem no longer applies. The importance of the JSCM principle has already been manifested by the vast amount of efforts devoted by researchers worldwide, resulting in numerous publications and proposed specific JSCM system implementations. We made our contribution to this field by developing not only a *general* JSCM scheme that works with most existing source-channel coder combinations without knowing their technical details, but also a *fast adaptive* JSCM scheme that has more practical value; we also extended the JSCM research field by introducing the concept of our new JSCM-based broadcast performance criterion: minimax disappointment. Our JSNM video communication research is another crucial step toward the direction of making JSCM research more complete, instead of being a mere extension.

Furthermore, we believe that the significance of JSCM research reaches beyond the development of specific systems and algorithms. It also addresses a fundamental question in designing a communication system: where should the designer put his research efforts? Should he focus on separate source and channel coder optimization (more efficient/robust source coders and more effective channel coders/network protocols), or should he instead focus on joint source-channel coder optimization and better cooperation (JSCM)? A good example would be the *heterogeneous packet flow* (HPF) protocol [4], which is an ambitious project whose ultimate objective is to replace the TCP protocol because of its superior ability in transmitting heterogeneous multimedia packets. Can smart protocols such as HPF remove the need to have a joint source-channel matching system, or make the gains insignificant? Answering these questions can help in designing effective communication systems.

CHAPTER 2

BACKGROUND KNOWLEDGE

In this chapter we give a brief overview of the basic background knowledge involved in designing a JSCM-based video communication system. We start by exploring the JSCM principle in greater depth, followed by a brief introduction of various source and channel coders, and finish by reviewing necessary background knowledge related to network protocols.

2.1 Joint Source-Channel Matching

Theorem 2.1 *(source-channel coding theorem): If $V_1, V_2, \ldots, V_n$ is a finite alphabet stochastic process that satisfies the asymptotic equality principle (AEP), then there exists a source channel code with $P_e^{(n)} \to 0$ if $H(\mathcal{V}) < C$.*

Conversely, for any stationary stochastic process, if $H(\mathcal{V}) < C$, the probability of error is bounded away from zero, and it is not possible to send the process over the channel with arbitrarily low probability of error.

The above Shannon's source-channel coding theorem [1] states that in designing a communications system, one can optimize the source and channel coders separately without sacrificing any overall performance. However, the validity of this theorem is contingent upon two important assumptions, which are generally not true in a realistic situation:

1. **Allowance of infinitely long codewords:** in other words, the system must be able to tolerate infinitely long delay. This is clearly not practical in a real-life situation, especially for video transmission, where even for low-quality requirement applications such as video conferencing (it has been determined that delays exceeding 150 ms do not give the viewer the impression of direct visual feedback).

2. **Point-to-point transmission path:** this is a fair assumption to make with a wireline transmission system, but certainly not with a wireless channel, where multi-path fading is unavoidable at times.

When either of these two assumptions is no longer valid, jointly optimizing the source and channel coders will yield performance gains (in this thesis we focus our efforts on the situation when the first assumption is invalidated), which leads to the *joint source-channel coding* (JSCC) [5] principle of designing a communications system.

From a general point of view, JSCC involves jointly optimizing the source and channel coders to obtain the optimal end-to-end transmission performance. A common approach is to jointly design the codewords for both the source and channel encoders, which effectively combines the source and channel coders into one source-channel coder; other approaches involve low-level tweaking of the coders, in essence a co-design of the source and channel coders. Here we give an incomplete list of work that has been done in the JSCC research area with regard to image and video transmission:

1. **Image Transmission.** The field of JSCC image transmission has been extensively covered by researchers: Davis and Danskin [6] described a joint source-channel allocation scheme for transmitting images losslessly over block erasure channels such as the Internet; Ramchandran et al. [7] studied multiresolution coding and transmission in a broadcasting scenario; Azami et al. [8] derived performance bounds for joint source-channel coding of uniform memoryless sources using binary decomposition; Belzer et al. [9] developed a joint source-channel image coding method using trellis-coded quantization and convolutional codes; Sherwood and Zeger [10] investigated unequal error protection for the binary symmetric channel; Man et al. [11] examined unequal error protection and source quantization; Lu et al. [12]developed a closed-form solution for progressive source-channel coding of images over bursty error channels; Fossorier et al. [13] studied joint source-channel image coding for a power constrained noisy channel.

2. **Video Transmission.** JSCC video transmission is investigated to a lesser extent primarily because of the much larger amount of data to process, more complex coders, and issues such as rate-control. Still, various approaches have been proposed: Bystrom and Modestino [14] investigated combined source-channel coding for video transmission over a slow-fading Rician channel; Lan and Tewfik [15] studied power-optimized mode selection for H.263 video coding in wireless communication; Zheng and Liu [16] used a subband modulation approach to transmit image and video over wireless channels; Xiong et al. [17] developed a progressive video coding scheme for noisy channels; Aramvith et al. [18]

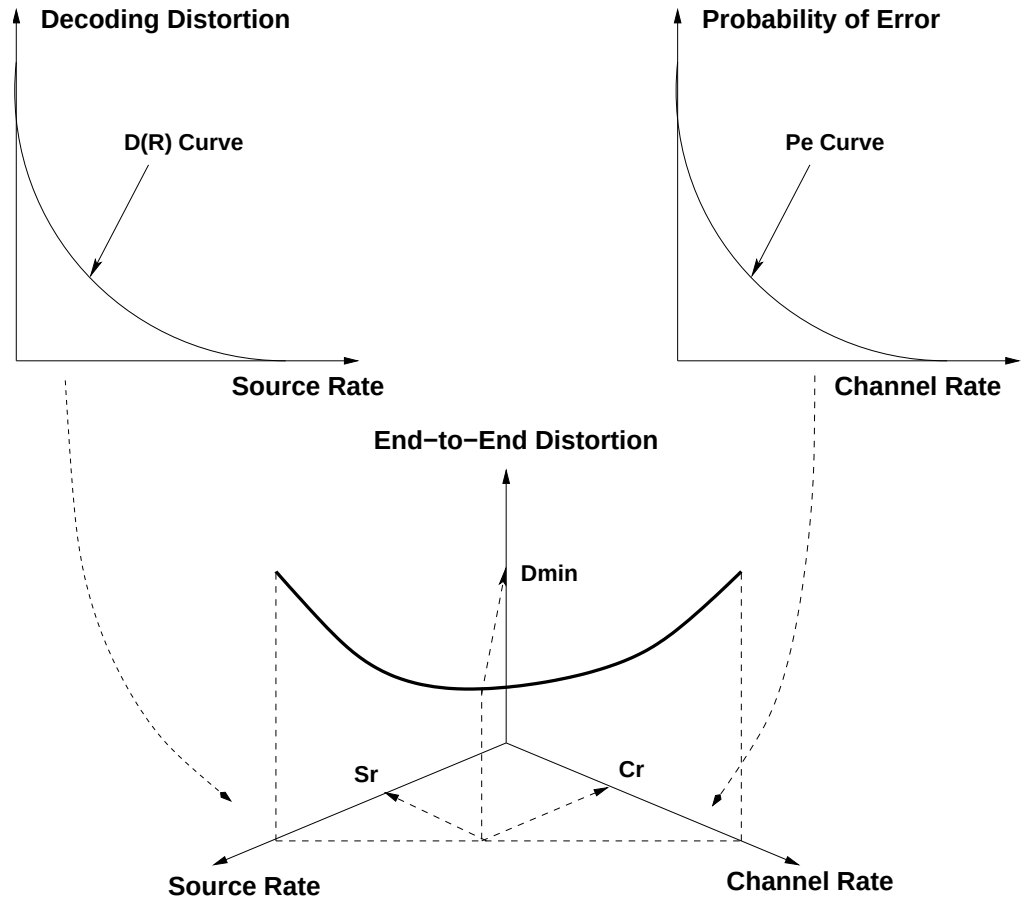

Figure 2.1 Joint source-channel matching (limited channel capacity).

proposed an alternative rate-control scheme for video transport over wireless channels using *automatic repeat request* (ARQ) retransmission scheme.

These approaches, although giving significant performance increase in general, usually suffer from a common disadvantage: they are optimized only for specific source-channel coder pairs, substituting either coder or both will almost certainly require a redesign of the entire transmission system. In other words, they do not possess enough degree of generality.

It is our desire to develop a truly general and effective JSCC system, which should perform well with a wide range of source-channel coder pairs, that leads to our specific choice of approach to realize JSCC: *joint source-channel matching* (JSCM). JSCM is a very simple but yet elegant concept, which can be illustrated using Figure 2.1:

1. The top left plot shows a typical *rate-distortion* (R-D) curve $D(R)$ that is conventionally used to characterize a video source encoder. It shows that for a *lossy* video encoder, if the

rate is increased, then more bits are used, the raw source stream is compressed less, and less distortion is incurred. Thus, given limited channel capacity, it is in our best interest to compress the video information at the highest rate possible. Most current video source coders are optimized to give the best performance (lowest decoding distortion) at a certain rate while assuming no channel transmission errors and that all the coded bits are received perfectly, which leads to the prolific use of sensitive data types such as marker bytes. There are efforts to develop channel-robust video source coders; for example, the *Moving Pictures Expert Group* (MPEG) video coding standard draft proposes various techniques such as data partitioning, *reversible variable length coding* (RVLC) and synchronization, etc., to reduce the compressed bitstream's sensitivity toward channel loss. However, they are still in development stage and not in wide use.

2. The top right plot shows a typical performance curve $P_e(R)$ of a channel encoder. It essentially shows that when we increase the channel rate (ratio of protection bits versus information bits), the probability of error P_e decreases. In other words, given sufficiently wide channel bandwidth, we should heavily protect all information bits before transmission to avoid channel errors and ensure that the end-to-end distortion could only result from lossy source compression. Similar to the source coders, most existing channel coders are designed for a specific channel and target bit error rate without explicit regard for the source coder characteristics.

3. The bottom plot shows a JSCM system with channel capacity limits: the entire bandwidth must be divided between the source and the channel coders. Allocating too much capacity to the source would cause an insufficient amount of channel protection and would lead to transmission errors, which decrease the end-to-end performance, whereas allocating too much capacity to the channel would result in an error-free transmission. However the source material is then over-compressed and thus yields a great distortion, which also decreases the end-to-end performance. There exists a fundamental trade off, or balance point, (S_r, C_r), which configures the system to give the optimal performance (smallest distortion D_{min}).

JSCM is not limited to channel-capacity allocation. It can also be used to match the source and channel coders in a power-limited situation, or whenever the source-channel coders share some system resources that are limited. In general, JSCM intelligently allocates limited system resources between the source and channel encoders to achieve the optimal end-to-end

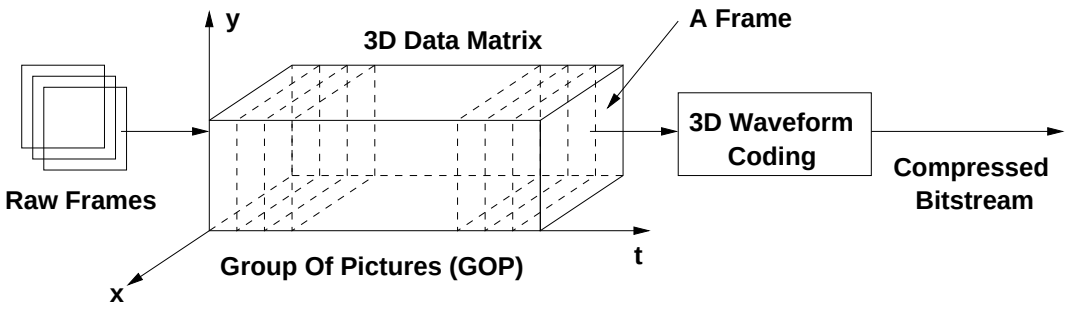

Figure 2.2 3D waveform video coding.

transmission performance. Some of our previous JSCM-related work can be found in [19] and [20].

2.2 Video Source Coding

It is necessary to have a basic understanding of various video source coders before one can successfully apply joint source-channel coding. Video coders are generally much more complex than image coders because of the extra temporal dimension and the high amount of data involved. Depending on whether or not temporal correlations are utilized, video coders can be roughly categorized into the two following major classes [21]:

1. Intraframe coding, which codes each video frame as an independent image using conventional image compression techniques, such as the *Joint Photographic Experts Group* (JPEG) standard [22], or the wavelet codec based on *set partitioning in hierarchical trees* (SPIHT) [23]. It has low coding efficiency and is only used when there is a limit on system complexity or a need for random access to individual frames.

2. Interframe coding, which exploits frame dependency. It can be further divided into the following subcategories:

    a. Three-dimensional waveform coding such as three-dimensional *discrete cosine transform* (DCT), shown in Figure 2.2.

    b. Motion-compensated coding such as the MPEG series and the H.26x series, as shown in Figure 2.3.

    c. Object/knowledge based model coding such as fractal video coding.

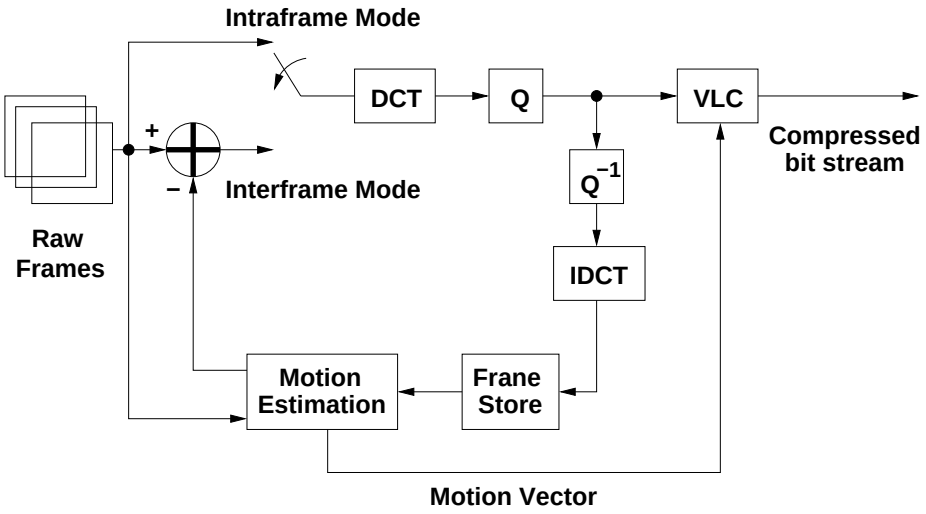

Figure 2.3 Motion-compensated video coding.

Here we list a few of the typical video coders from each of the above categories. For details of their encoding/decoding procedures and rate-distortion characteristic estimation, please refer to Appendix A.

2.2.1 Motion-JPEG

Perhaps the simplest video coder from the intra-frame coding category and the default codec for low-cost video conferencing [24], [25]. Motion-JPEG simply codes each frame using the highly efficient JPEG format and transmits the coded frames sequentially. In spite of its simplicity, it is commonly used because of the relatively uniform frame quality, its ability to start decoding at any particular frame, and no frame-to-frame error propagation. In addition, there are hardware chip-implementations of Motion-JPEG codec [26], [27] for various platforms such as Sun Solaris. It also has the nice property that the rate-distortion function remains approximately the same for video frames in the same sequence given there is no scene change; this property greatly simplifies the estimation of R-D characteristics for the source coder, since we only have to perform the estimation procedure once for every statistically distinctive sequence. For a typical R-D curve of the Motion-JPEG encoder and its estimation procedure, see Appendix A.1.

Although it is simple and hardware-implementable, Motion-JPEG is not the ideal choice for JSCM. The reason lies in JPEG's extensive use of error-sensitive marker bytes. Those markers help to achieve better compression ratio, but at the same increase the output stream's suscep-

tibility to channel errors. A corrupted marker byte could significantly affect the quality of the decoded frame or, under certain circumstances, render the entire received frame undecodable.

To address this error-sensitivity issue, researchers have proposed approaches such as transcoding or collecting all the error-sensitive marker bytes and transmit them with heavy protection [28]. These approaches, though effective to some extent, could be inefficient and computationally expensive. In this thesis, we instead choose the "get-it-or-lose-it" approach and try to minimize frame losses and compensate for the lost frames by using error-concealment techniques. For details, refer to Appendix B.

2.2.2 Motion-SPIHT

Motion-SPIHT is another candidate from the intraframe coding category; it replaces the JPEG coder in Motion-JPEG with the wavelet-based SPIHT image coder. SPIHT has excellent progressive properties, with a highly prioritized output bit stream that can be truncated at any location and yet still produces a coarse version of the original, which facilitates the implementation of *unequal error protection* (UEP). Its framewise rate-distortion functions also have the same frame-independence property as that of Motion-JPEG.

One advantage SPIHT has over JPEG as a still frame encoder is that it does not contain any special important marker bytes as JPEG does. Every correctly-received bit could introduce extra gain in performance. In a error-prone situation, Motion-SPIHT could be perform much better than Motion-JPEG in terms of end-to-end distortion.

2.2.3 Conditional block replenishment

Conditional block replenishment (CBR) coder is one of the earliest approaches in interframe coding (in fact, both H.26x and the MPEG series can degenerate to CBR in their most primitive mode) and has a wide usage in videoconferencing applications, where the frames consist of mostly still background and not much movement, and in multicast systems such as *multicast backbone* (M-Bone) on the Internet. It has certain advantages, such as coding simplicity and substantial robustness to transmission errors, over motion-compensation-based video coders like MPEG. A CBR coder works in the following fashion:

1. The first frame is encoded using conventional image compression techniques (such as JPEG) and transmitted. It is denoted as the *index* frame.

2. The following frame is divided into blocks and then segmented into "modified" and "unmodified" blocks with respect to the *previously transmitted* frame (not the previous raw frame). The "modified" blocks can be defined as those blocks whose *mean square error* (MSE) with regard to the previously transmitted frame are greater than a certain threshold, or simply the K blocks whose MSE are greatest. Other alternative criteria can also be used.

3. Block address, luminance and chrominance information in the "modified" region are then encoded by means of any *differential pulse coded modulation* (DPCM) method and transmitted.

4. The following frames are treated in the same fashion until a forced frame replenishment occurs, at which point the next frame is transmitted as an index frame.

CBR has the intrinsic disadvantage common to all conditional replenishment coders, which is error propagation. A corrupted block resulted from transmission errors will propagate to all the subsequent frames until the next forced frame replenishment occurs (for example, if a black block appears in a frame, it usually stays there until the next frame replenishment). Thus, the frame replenishment frequency is a crucial parameter in maintaining playback quality.

Conditional replenishment can also be performed in the frequency domain or on a frame basis [29]. A review of conditional replenishment algorithms can be found in [30].

2.2.4 3D-SPIHT

The 3D-SPIHT format [31], [32] is an extension to the 2D-SPIHT image compression format; similar to 2D-SPIHT, it has the feature of being *embedded* or *progressive*, which means that the bits in the compressed stream are ordered by their relative importance. Unlike 2D-SPIHT, whose progressive property is within each frame only, 3D-SPIHT produces progressive bitstreams for each *group of pictures* (GOP). An obvious advantage of using an embedded coder is that if the receiver has failed to obtain the tail portion of the bitstream, it could still use what it has received to reconstruct a version of the original with slightly inferior quality; generally, any extra bit received increases the received quality.

However, since 3D-SPIHT achieves high compression ratio by utilizing both the spatial and temporal redundancy in a video sequence, it gives only mediocre results when the video sequence contains a high amount of motion information (less spatial and temporal redundancy). Thus,

theoretically it is only suitable for applications such as videoconferencing and video-phone, and requires motion-compensation extensions to perform well for fast-motion video sequences [33]. In simulations we observe that for relatively low-resolution sequences, such as the *quarter common intermediate format* (QCIF), 3D-SPIHT has H.26x-comparable performance even when significant amount of motion exists. The 3D-SPIHT format is not commonly used in practical applications yet because of its high computation requirement. However, with the advancement of digital signal processing algorithms and faster hardware chips, the computation requirement barrier is eventually going to dissolve and coders such as 3D-SPIHT would enter the commercial application ground.

2.2.5 H.26x

Unlike the previously mentioned video coders, the H.26x family of video coders (H.261, H.263, H.263+) is based on advanced motion-compensation compression techniques and targets mainly low-rate applications such as videoconferencing and videophone, where the input frames consist of mostly background information and the amount of motion is limited. Some of its compression techniques were later adopted and refined by the MPEG-1 and MPEG-2 video compression standards. The latest MPEG-4 standard is basically an H.263 kernel wrapped with a shape-coder layer.

The H.26x encoding procedure could be roughly described as the following: first a reference frame (I-frame) is compressed using the traditional image compression method, then motion-vectors for the next frame with regard to the *reconstructed* reference frame are estimated and encoded. H.26x coders are amenable to low-cost *very large scale integration* (VLSI) circuit implementation, which is rather important for widespread commercialization of videophone and teleconferencing applications. For details of the H.26x encoders, please refer to Appendix A.5.

2.2.6 MPEG 1/2/4

The MPEG video compression series (MPEG1, MPEG2, MPEG4), developed by the International Organization for Standardization (ISO) committee, has become the de facto standard for video storage. MPEG-1, approved as an international standard in 1992, has been developed for storage of *common intermediate format* (CIF) video at about 1.5 Mb/s on various media; MPEG-2 was designed for data rates of up to 20 Mb/s for high-definition video; the on-going

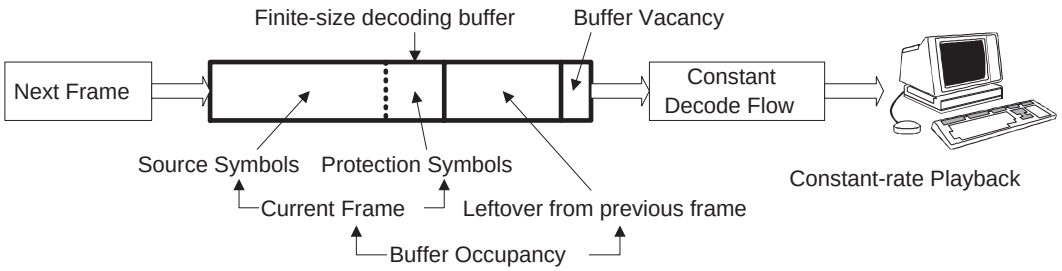

Figure 2.4 Rate-control for video coders.

MPEG-4 development, however, concerns itself mainly with very-low-bitrate compression (8-32 kb/s) for videophone applications.

The MPEG series and the H.26x series share several key components in compression techniques, for example, block-based motion compensation, DCT transform, and entropy coding. Their differences are mostly the result of different target bitrates, and are significant only in their respective advanced compression profiles. For details of the MPEG series, refer to the ISO MPEG standards.

The MPEG video compression series are traditionally difficult to model mathematically because of the block-based approach and motion-compensation techniques, thus, real-time estimation methods are often used [34].

2.3 Rate Control

Rate (buffer) control (demonstrated in Figure 2.4) is a key issue in video transmission [35] [36], since a small, finite delay between transmission of frames must be obtained. Furthermore, in a practical implementation the decoder usually has a finite decoding buffer-size and a constant decode-flow.

Suppose we have a video frame sequence divided into GOPs of size N. Each picture or frame is coded to have a stream length of $s(i, q_i)$ in bits as defined previously. For the transmission of the ith frame, a certain amount of protection, denoted by P_i, is used. Then the *buffer occupancy*, b_i, after frame i is transmitted can be expressed as

$$b_i = \max(b_{i-1} + s(i, q_i) + P_i - R, 0), \ i = 1, 2, \ldots \tag{2.1}$$

where R is the constant decoding stream flow (stuffing-bits will be added to avoid underflow when b_i is smaller than zero). The buffer size constraint is then represented as

$$b_i \leq b_{max},\ i = 1, 2, \ldots \tag{2.2}$$

where b_{max} is the finite decoding buffer size. The optimization thus becomes a constrained problem, which usually requires techniques such as the Lagrange algorithm or the penalty function to solve.

Among all the video coders we mentioned in the previous section, most require a rate-control mechanism except the 3D-SPIHT coder, in which case we allow an overall delay of one GOP and start playback only after the entire GOP is received.

2.4 Channel Coding

Here we give a brief introduction to two typical widely used channel coders: the RS codes and the RCPC codes. Note that both these two codecs attempts to correct any errors present but do *not* guarantee an error-free output. For example, an RS codeword might be corrupted by so many errors that it becomes another valid RS codeword, the decoder would simply assume that no errors are present. Thus, these codecs are usually used in conjunction with parity-checking mechanisms to ensure a valid output.

2.4.1 Reed-Solomon codes

RS codes [37] are a well-known class of block codes with good error-correction properties. They have excellent abilities to correct channel burst errors, which are common in a wireless environment. An RS code defined by (n, k, t) is a length-n code that contains $k = n - 2t$ source symbols, $2t$ protection symbols, and can correct t symbol errors. There are RS codes for various n, most commonly $n = 2^m - 1$ where m is the symbol length in bits (in our scheme we will use $m = 8$, because we usually access frames in bytes). An (n, k, t) code will be unable to recover the original k data symbols if more than t errors occur.

In our JSCM schemes, we employ a packet-based approach in RS channel coding, in which case each transmission packet is an RS coding unit and contains both the source symbols and protection symbols. We define, for video transmission, the *frame transmission success*

probability P_{fsucc} as

$$P_{fsucc} = \prod_{n=1}^{M} P_{psucc} \tag{2.3}$$

$$P_{psucc} = \sum_{k=0}^{p/2} \binom{L}{k} {P_{es}}^{k} {P_{ss}}^{L-k} \tag{2.4}$$

$$P_{ss} = 1 - (1 - P_{eb})^{m} \tag{2.5}$$

$$P_{eb} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{2.6}$$

where P_{psucc} is the *packet transmission success probability*, M is the total number of packets to transmit, P_{ss} is the *symbol transmission success probability*, L is the packet length, p is the number of protection symbols in each packet, P_{eb} is the bit-error probability, m is the number of bits per symbol, E_b is the fixed power per bit, and N_0 is the channel noise variance. In order for the above equations to be valid, sufficient interleaving must be employed so that the independence assumption is satisfied.

2.4.2 Rate-compatible punctured convolutional codes

RCPC codes, introduced by Hagenauer and colleagues [38], [39], [40], extend traditional convolutional codes by puncturing a low-rate $1/N$ code periodically with period P to obtain a family of codes with rate $P/(P+l)$, where l can be varied between 1 and $(N-1)P$. The *rate-compatible* restriction on the puncturing tables ensures that all code bits of high rate codes are used by the lower-rate codes, thus allowing transmission of incremental redundancy in *forward error correction* (FEC) schemes and continuous rate variation to change from low to high error protection within a data frame. RCPC codes are good for source-channel coding because the basic structure of the codec remains the same as the coding rate changes.

2.5 Network Channels

As a fundamental part of network systems, network protocols allow one to specify or understand communication without knowing the details of the physical network architecture [41]. Multiple network protocols exist in today's network systems to deal with the seemingly overwhelming set of transmission problems such as congestion, routing, data protection, etc. Each protocol has its own advantages and disadvantages with regard to transmitting video packets.

Here below we briefly discuss several of the most widely-used protocol candidates, focusing on their individual advantages and disadvantages for applying JSNM video transmission.

2.5.1 User Datagram Protocol

The User Datagram Protocol (UDP) [41], [42], [RFC 768], based on the underlying *Internet Protocol* (IP) [RFC 791] [RFC 1122], provides an *unreliable, connection-less* delivery mechanism to transmit datagrams over the network. It does not use acknowledgments, nor does it provide feedback to control the rate at which information flows. The application designer assumes full responsibility for handling the problem of reliability, including packet loss, duplication, delay, and out-of-order delivery.

The UDP protocol has certain advantages in delivering video packets; because it has no feedback mechanism, there is no extra delay caused by the receiver requesting retransmission. Also, UDP has a relatively small overhead; thus, it is possible to custom-build protocols on top of UDP that are tuned to the task of transmitting video packets.

The unreliable nature of the UDP protocol, however, also poses a problem for transmitting video packets that contain crucial information bits such as marker bytes. The loss of such packets usually causes disastrous effect on the receiver end, often resulting in total frame loss. Thus, the UDP protocol is mostly useful for transmitting progressive video streams, where heavy penalty for packet loss can be avoided by applying unequal error protection. Besides, UDP does not have a congestion-control mechanism, which makes it more practical in a local network environment than in a larger hybrid network such as the Internet.

2.5.2 Transmission Control Protocol

The Transmission Control Protocol (TCP) [41], [42], [RFC 793] provides reliable stream delivery of packets over networks. TCP uses positive acknowledgement with retransmission techniques to provide reliability. It also has built-in congestion-control algorithms, a timer for network time-out, packet sequence numbers for out-of-order packets, and buffered transfer based on the sliding-window concept [RFC 813].

The TCP protocol has a lot of appealing properties suitable for transmitting video packets. For example, it is a reliable delivery service, so video streams that are not progressive can be transmitted without the worry that some crucial information packet might be lost; it has full-duplex connection via which the receiver can feed back information to facilitate the sender's

rate-control process; it also provides a push mechanism for immediate delivery and the ability to transmit out-of-band data, which is highly useful in sending control commands to the receiver; it assigns sequence numbers to its packets, so packet repackaging at the receiver end is done inherently.

The major disadvantage of using TCP to transmit video packets arises from the potentially intolerable amount of delay it might introduce; especially in a highly congested or error-prone network, the receiver would have to wait for a long period of time just for a certain packet to arrive intact. It is inherently unacceptable for video communication sessions, especially videophone and videoconferencing applications. Also, because of the transmission delay, one might question the credibility of the channel information feedback sent by the receiver.

2.5.3 Real-Time Transfer Protocol

The *Real-Time Transfer Protocol* (RTP) [RFC 1889] provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast networks. It is closely coupled with a control protocol such as *Real-Time Transfer Control Protocol* (RTCP) to allow monitoring of the data delivery, and to provide minimal control and identification functionality.

RTP is typically run on top of UDP to make use of its multiplexing and checksum services; both contribute parts of the transport functionality. It can also be used with other suitable underlying network or transport protocols.

RTP is suitable for network video communication because it provides services such as payload type identification, sequence numbering, timestamping, congestion control and delivery monitoring. However, RTP itself does not provide any mechanism to ensure timely delivery or provide quality-of-service guarantees, but relies on lower-layer services to do so. It represents a new style of protocol following the principle of application-level framing and integrated layer processing. It is a protocol framework that is deliberately incomplete.

It is because of the incompleteness, and the resulting flexibility, that we believe an RTP-like protocol may be the best choice for a joint source-network video communication system.

2.5.4 Heterogeneous Packet Flow Protocol

The HPF [43] protocol is an ambitious new transport protocol, designed by the TIMELY group at the University of Illinois at Urbana-Champaign for effectively supporting hybrid data packet flows in the internet environment.

HPF has the following key features:

- HPF supports packet flows that contain different quality-of-service requirements such as reliability, priority, and deadlines.
- HPF supports application-level framing.
- HPF enables the use of application-specified priorities as hints for network routers to do preferential dropping during congestion.
- HPF decouples the congestion control and reliability mechanisms in order to support congestion control for unreliable and heterogeneous packet flows.

HPF differs from RTP in that although both protocols let the application specify policies for framing, reliability, timing and priority, HPF insists on a clean separation between policies and mechanisms. In other words, HPF believes that the transport protocol alone should provide the mechanisms without the participation of the application.

Interestingly, this concept of clean separation between policies and mechanisms goes directly against the joint source-channel coding principle, which requires a joint optimization and close integration of the source and channel coders. In the case of an HPF-based system, the source coder simply assigns priority, cost, and deadline information to the packets without utilizing channel information, and it is then up to the channel coder (protocol) to carry out procedures such as packet delivery and dropping. There is no *joint* optimization or bidirectional information exchange between the source and channel coders. It would be of great importance to find out which theory leads to a better video communication system. Besides using HPF as a comparison in our research, it would also be interesting to discover whether or not JSCM can bring further performance gains to an HPF-based system.

Here we conclude our background-information chapter, in the following chapters we discuss thoroughly the application of JSCM to video communication, starting with the most fundamental case: peer video transmission.

CHAPTER 3

JSCM PEER VIDEO TRANSMISSION

In this chapter we present the foundation of our entire JSCM research: the application of JSCM for peer video transmission over point-to-point channels. In our earlier work, we have developed a general JSCM framework suitable for cases where simple video coders are used (i.e, real-time rate-distortion characteristics estimation is possible) [44]. In this thesis, we extend the general system and propose an adaptive matching scheme for complex video coders. We demonstrate the effectiveness of our schemes via simulations using several source-channel coder combinations.

3.1 The Underlying JSCM Problem

The underlying JSCM problem can be illustrated using Figure 3.1:

At the sender end, a series of raw video frames are source-channel encoded and transmitted over the channel (where errors might occur to corrupt the information) to the receiver; at the receiver end, corresponding channel-source decoding is performed to obtain an estimated version of the original raw video frames. Our objective is to minimize the *expected* end-to-end distortion between these two sets of frames. The JSCM operation intelligently allocates limited system resources (channel capacity, transmission power, etc.) to the source and channel coders, based

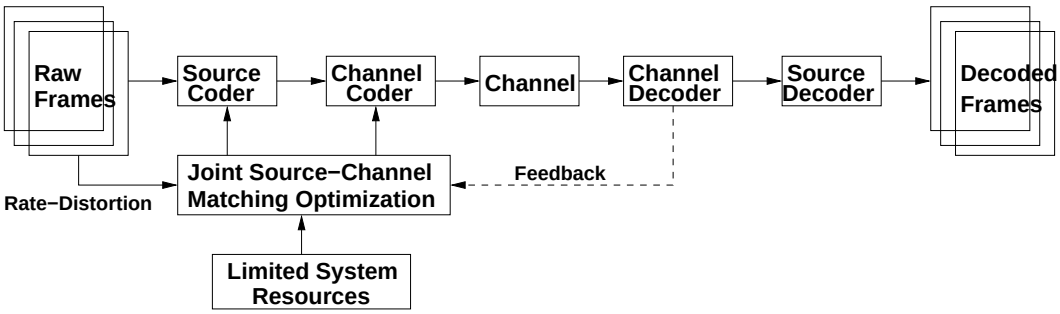

Figure 3.1 The underlying JSCM problem.

on the rate-distortion characteristics of the source, and the accurate current channel situation estimate obtained from a separate reliable feedback channel (both the existence of the feedback channel and the accuracy of feedback information are crucial to the JSCM system, which is in later sections).

Mathematically, the JSCM problem can be expressed as the following (using the channel capacity-limited case and MSE distortion as an example):

$$
\begin{aligned}
Y &= \mathcal{S}_e(X) && (3.1)\\
Z &= \mathcal{C}_e(Y) && (3.2)\\
\hat{Z} &= \mathcal{H}(Z) < C && (3.3)\\
\hat{Y} &= \mathcal{C}_d(\hat{Z}) && (3.4)\\
\hat{X} &= \mathcal{S}_d(\hat{Y}) && (3.5)\\
J &= \mathbf{E}\ (X - \hat{X})^2 && (3.6)
\end{aligned}
$$

where X denotes the original raw video frames, which are encoded using source encoder $\mathcal{S}_e$ to obtain Y; Y is further encoded using channel encoder $\mathcal{C}_e$ to obtain Z, which cannot exceed the channel capacity limit C; the channel transmission process $\mathcal{H}$ maps Z to $\hat{Z}$, which is subsequently passed through the channel decoder $\mathcal{C}_d$ and source decoder $\mathcal{S}_d$ to obtain the final decoded video frames $\hat{X}$; and J is defined as the MSE between X and $\hat{X}$. JSCM sets out to find the optimal $\mathcal{S}_e$ and $\mathcal{C}_e$ which minimize J without invalidating the capacity constraint.

3.2 JSCM General Framework

Most general JSCM frameworks, we believe, should achieve the following objectives:

1. Obtain the best end-to-end overall system performance and satisfy the constraints posed by limited channel capacity, small allowable delay, and finite decoding buffer-size (rate control).

2. Achieve a certain level of generality so that it can be applied to various standard source-channel coder pairs without relying on detailed low-level technical knowledge about the coders.

3. Can be implemented in real-time with reasonable complexity requirements.

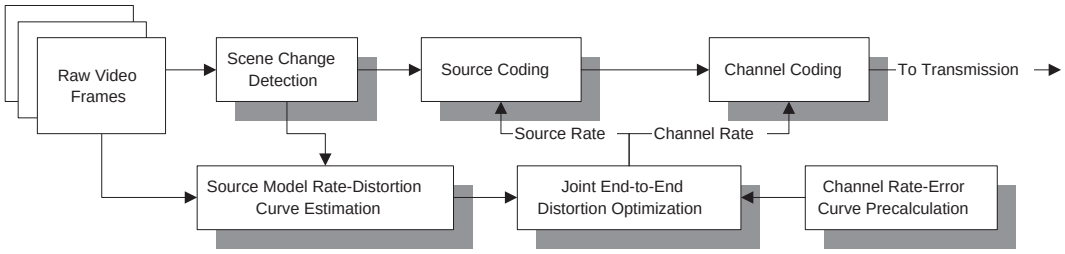

Figure 3.2 General joint source-channel matching system.

We have demonstrated, via various simulations, that we have achieved all the three objectives above. The resulting general JSCM system is shown in Figure 3.2, which contains six functional blocks:

1. **Source rate-distortion model estimation**: This block models the source coder as a black box with rate-adjusting dials and estimates its rate-distortion characteristics in real-time (for cases when this operation is too expensive, we revert to the adaptive JSCM scheme in Section 3.3.)

2. **Channel model precalculation**: The channel encoder's performance curve (probability of error versus rate) can usually be either calculated from analytical expressions (RS codes) or estimated from offline simulations (RCPC codes). This block performs this precalculation step and stores the tabularized values for use in the JSCM optimization.

3. **JSCM optimization**: This block solves the constrained optimization problem by applying techniques such as penalty functions and gradient-descent algorithms. The solution provides the source and channel encoders with their individual coding configuration parameters.

4. **Source coding**: This block performs source-encoding on the raw video frames, based on the parameters passed from the JSCM block.

5. **Channel coding**: This block performs channel-encoding on the source-encoded bitstream, based on the parameters passed from the JSCM block.

6. **Scene-change detection**: This block is required because we observe in simulations that the source coder's rate-distortion characteristic curve is data-dependent. In video sequences, a scene-change usually accompanies the change of the statistical properties of

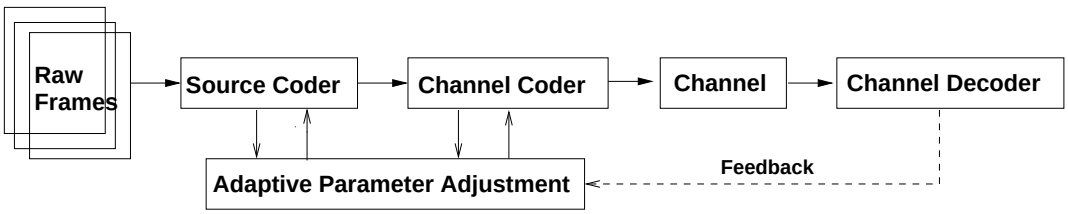

Figure 3.3 Adaptive joint source-channel matching system.

the current video sequences, thus requiring a re-estimation of the rate-distortion characteristic curve.

Our proposed general JSCM system proves to work well with a variety of source-channel coding pairs, simulation results are presented in Section 3.4.

3.3 Adaptive JSCM Scheme

In the above-described JSCM system, we make the assumption that it is possible to obtain the source rate-distortion characteristics using real-time estimation techniques. This assumption is not always valid, given the complexity of some of the current video encoders such as the H.26x series and the MPEG series. For the cases when this assumption is invalidated, we propose an alternative adaptive JSCM scheme. It attempts to converge to the locally optimal system configuration by changing the coding parameters on-the-fly based on feedback information and steers the system toward the direction of better performance. In the case of a constant or very slowly varying channel, it converges to the locally optimal system configuration; in the case of a slowly-varying channel, it dynamically tracks the channel variation and provides a sometimes suboptimal configuration. The speed of convergence is determined by how aggressively the system adjusts the source-channel coding parameters. The system is illustrated in Figure 3.3.

The system is similar to the general JSCM system shown in Figure 3.2. However, the *rate-distortion estimation* block is removed because of its high computational requirement with a complex video coder, and the *joint source-channel matching* block is replaced with the *adaptive parameter adjustment* component, which performs the adaptation task, instead of attempting to obtain a general solution to the optimization problem.

Suppose we use S_r to denote the source encoding rate, C_r to denote the channel protection rate, and P_e to denote the channel error rate. the basic adaptation strategy can be described as the following:

Initialize: $S_r = S_1$, $C_r = C_1$
If $P_e < P_o + \epsilon$ or $P_e < P_o - \epsilon$
we are in the stable zone
$S_r = S_o$ and $C_r = C_o$
else if $P_e < P_o - \epsilon$
we are in forward adaptation
$C_r = C_o - \Delta C_r$
$S_r = C - C_r$
else if $P_e > P_o + \epsilon$
we are in backward adaptation
$C_r = C_o + \Delta C_r$
$S_r = C - C_r$
endif

where S_1 and C_1 are the preset initial values for S_r and C_r; C is the total channel capacity; ϵ is the stable-zone threshold; S_o, C_o, and P_o are the original values for S_r, C_r and P_e; and ΔC_r is the step size used when modifying channel protection rate.

1. **Initialization**: In the initialization stage, the system assigns the source and channel coders a set of common coding parameters.

2. **Forward adaptation**: The system enters *forward adaptation* stage when the channel situation improves. The feedback information informs the JSCM parameter adjustment component that the channel has improved, and the JSCM block reduces the resources consumed by the channel coder (channel rate, for example) by a small step (the step size can also be adaptively determined); correspondingly, more resources are then allocated to the source coder, end-to-end performance is expected to be improved. The forward adaptation stage continues until the system enters the *stable zone* or *backward adaptation* stage.

3. **Backward adaptation**: The system enters *backward adaptation* stage when the channel situation deteriorates. The JSCM parameter adjusting block then increases the resources

consumed by the channel, also by a small step; correspondingly, fewer resources are allocated to the source encoder. However, more allocated channel resources prevent channel errors, and the combined effect is expected to increase the end-to-end performance. The backward adaptation stage continues until the system enters the *stable zone* or *forward adaptation* stage.

4. **Stable zone**: Presumably the system could oscillate between forward and backward adaptation states due to small variations in the channel situation, causing stability problems and unnecessarily frequent coding parameter changes (which could be costly in a hardware implementation). To alleviate this problem, we claim that the system has entered the *stable zone* when the change in the channel error rate is below a certain threshold, in that case, the source-channel coding parameters are preserved from the previous state. The system leaves the *stable zone* when the change in the channel error rate is greater than the preset threshold ϵ.

The adaptive JSCM system works as expected with complex video encoders such as H.263. The rate of convergence can be adjusted by modifying the step size ΔC_r. Furthermore, we discover from simulation that the adaptive system converges to the general system as long as the rate of channel variation (defined as the time duration required for a 10% change in P_e) is longer than twice the feedback interval. Simulation results are presented in Section 3.4.

3.4 Simulation Results

In this section we present the simulation results for our adaptive matching system. For completeness, we also first revisit the simulation results for the general matching system, which was developed in our earlier work.

3.4.1 JSCM general system

To demonstrate the effectiveness of our proposed JSCM general system, we use the following simulation parameters. (For details of specific system implementation details, please refer to Appendix B.)

1. **Simulation video sequence**: The standard fast-motion CIF format grayscale "Football" sequence (for other simulation sequences such as "Miss America" and "Table tennis," results are similar and not presented here) with a GOP structure of size 20.

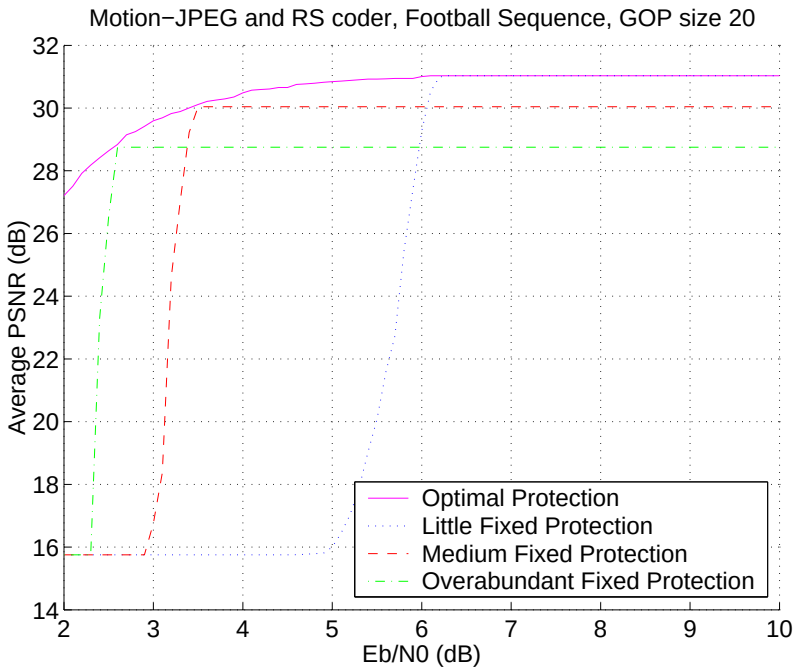

Figure 3.4 Simulation results for a general JSCM video peer-transmission system in a rate-constrained situation, using Motion-JPEG video source coder and RS channel coder.

2. **Video source coder**: We tested both Motion-JPEG and CBR coders.

3. **Channel coder**: Reed-Solomon block codes with codeword length 255.

4. **Transmission channel**: Wireless channel with *additive white Gaussian noise* (AWGN), *signal-to-noise ratio* (SNR) ranging from 2 to 10 dB (for the Motion-JPEG case) and from 2 dB to 7 dB (for the conditional block replenishment (CBR) case). We assume slowly-varying channels and accurate feedback information.

The simulation results (Figure 3.4, 3.5, and 3.6) show the following when we compare the performance between JSCM and non-JSCM systems:

- **No JSCM**: Fixed source compression and channel protection rate parameters.

    1. If the system chooses a fixed parameter set where the channel protection rate is relatively low (higher source rate), the system behaves well in the region of high channel-SNR but fails miserably when channel SNR deteriorates below a certain level (shown in Figures 3.4 and 3.5 as the dotted curves).

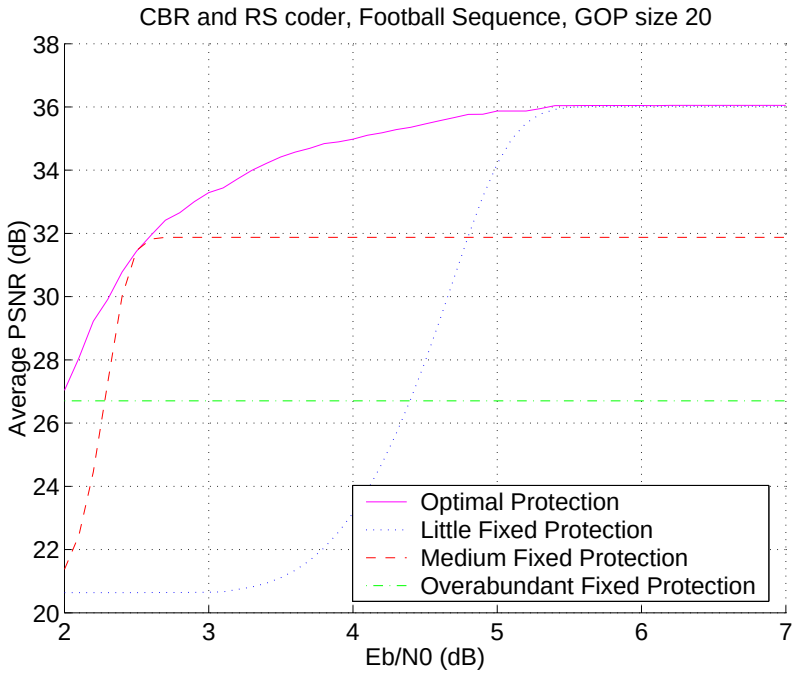

Figure 3.5 Simulation results for a general JSCM video peer-transmission system in a rate-constrained situation, using CBR video source coder and RS channel coder.

2. If they system chooses a fixed parameter set where the channel protection rate is relatively high (lower source rate), the system behaves well over a wide range of channel-SNR values, but in the region of high channel-SNR it fails to capture a possible performance increase of 3 dB because it cannot exploit good channel conditions (shown in Figures 3.4 and 3.5 as the dash-dot curves).

3. If the channel protection rate is chosen to be a medium value, then the system still has the two problems mentioned above in the regions of low channel-SNR and high channel-SNR, albeit to a lesser extent (shown in Figures 3.4 and 3.5 as the dashed curves).

- **JSCM**: Source compression and channel protection rate are jointly optimized by the JSCM procedure to obtain best end-to-end results.

  The JSCM performance curve stays above those of the fixed-parameter systems and always achieves the highest end-to-end *peak signal-to-noise ratio* (PSNR). It is essentially the convex hull of all the fixed-parameter system performance curves, with each point corresponding to an optimal configuration for a particular channel SNR (shown in Figures 3.4 and 3.5 as the solid curves on top of the other curves).

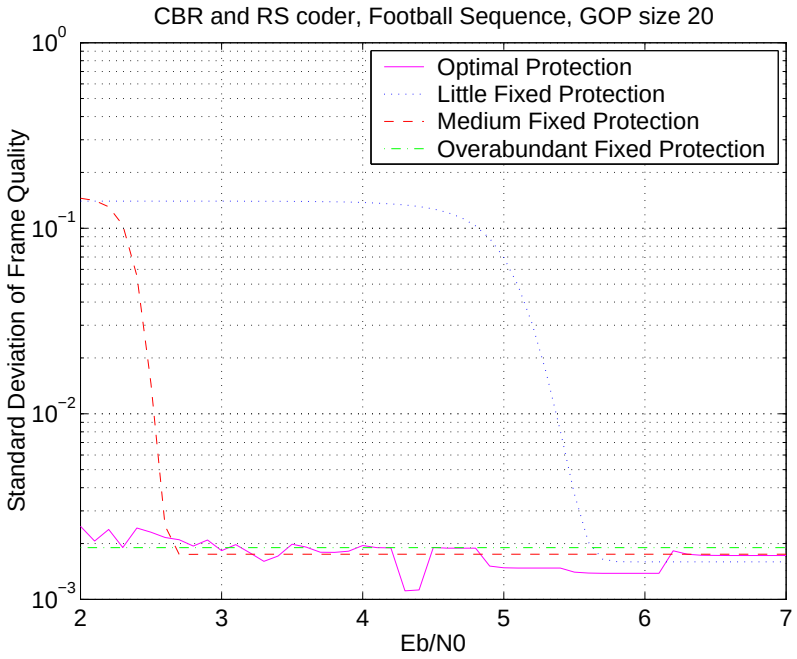

Figure 3.6 Simulation results for a general JSCM video peer-transmission system in a rate-constrained situation, using CBR video source coder and RS channel coder. Frame quality variation is used as an alternative performance criterion.

- **Alternative performance criterion**: In order to demonstrate the generality of our JSCM system, we use the frame quality variation factor in addition to the average frame PSNR as an end-to-end performance measure. From the result plot (Figure 3.6) we can observe that the JSCM performance curve does a better job of keeping the frame quality variation low than either the little-protection or the overabundant-protection case. Its performance is comparable to that of the medium-protection case in most portions of the entire channel SNR range, but yields a significantly lower variation in certain regions.

It is now clear why the implementation of JSCM requires the existence of a separate feedback channel that provides an accurate up-to-date channel condition estimate. JSCM relies on the knowledge of current channel condition to determine the appropriate amount of channel protection to employ and is highly sensitive to the accuracy of the feedback information. For example, in Figure 3.4, if the channel SNR is 5 dB and the feedback erroneously reports it to be at 6 dB, the performance loss caused by the incorrect coding parameter decision from JSCM could be more than 15 dB. Thus, it is imperative to have an accurate estimation of the current channel condition for JSCM to function.

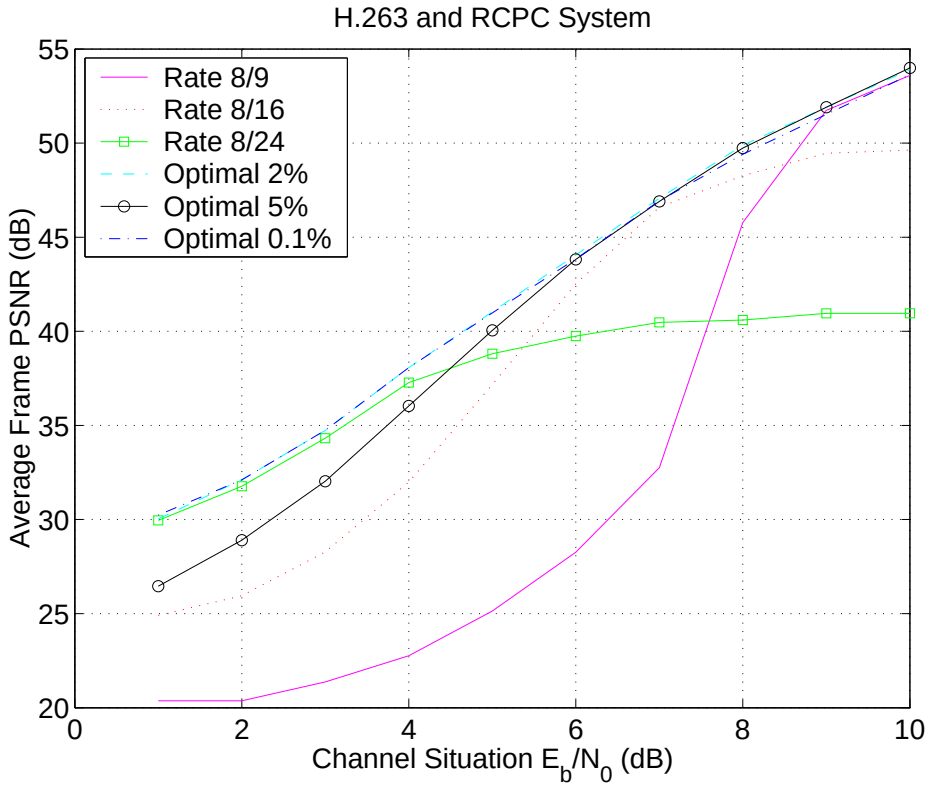

Figure 3.7 Simulation results for an adaptive joint source-channel matching video peer-transmission system in a rate-constrained situation, using H.263 video source coder and RCPC channel coder. Percent values in figure are corresponding thresholds, ϵ.

3.4.2 JSCM adaptive system

We chose the H.263 video encoder and RCPC channel coder to demonstrate the adaptive JSCM system. The corresponding coding parameters are H.263's *MQUANT* factor, modified on a macroblock basis, and the RCPC's puncture rate. The encoded information bit stream is packetized and sent over the channel, and the channel decoder sends back the *cumulative packet loss ratio* as an estimate of the current channel situation. More details about the system implementation can be found in Appendix B.

From the result in Figure 3.7 (each percent value corresponds to a different stable-zone threshold ϵ) we essentially observe the same behavior as with the general JSCM system: the fixed-parameter system's performance drops as channel deteriorates, and stays constant when the channel improves greatly, while the JSCM system's performance curve always resides above those of the fixed-parameter systems.

3.4.3 Summary

We summarize our findings and present them in the form of two conjectures, which also serve as an underlying supporting factor for the next chapter on JSCM broadcasting. We emphasize in the conjectures the fact that the "optimal performance" for a user is defined for the current channel condition only.

Conjecture 3.1 *The optimal video transmission performance for a user, with regard to the current channel condition, can be obtained by applying the JSCM principle in a JSCM-capable system toward this particular user.*

Conjecture 3.2 *The adaptive JSCM system converges to the general JSCM system (within a threshold) given that the speed of channel variation (defined as the time duration for a* 10% *change in* P_e*) is longer than twice the feedback interval.*

CHAPTER 4

BROADCAST: MINIMAX DISAPPOINTMENT

Users of media broadcasting services generally experience different grades of overall performance because of the nonuniform quality of their channel conditions. In order to provide all users with equally satisfactory performance, we need to define a meaningful and fair performance measure for an overall broadcasting scheme. In this chapter we first introduce the most general form of the broadcasting performance criterion, and then, based on its infinite-norm special case, we propose an alternative performance criterion named "minimax disappointment." This criterion utilizes the concept of layered service levels and the basic principle of JSCM; it minimizes for all users the maximum value of performance degradation between the received performance and the users' optimal expectation, given each user's individual channel situation. At the end of this Chapter we develop several demonstration broadcasting systems and a gradient-based optimization scheme.

4.1 Motivation for an Alternative Criterion

In a broadcasting situation, the central broadcasting station transmits the same set of signals to all users of its service without regard to their individual connection to the station. Received performance for different users may vary because of different channel qualities. A typical example would be a simple wireless broadcasting scenario, where users are scattered in a pattern of co-centered circles of different radii (shown in Figure 4.1). Depending on their respective distances to the station and other factors such as geographical environment, structure density, local weather, and electromagnetic noise activity, users can be categorized into different classes which observe different channel SNR and fading profiles. In general, the farther away the user is from the station, the worse the transmission quality because of signal energy decay over distance.

In view of the above situation, it is considered good practice to define multiple user classes and provide layered levels of service. In other words, the broadcast signal is divided into multiple

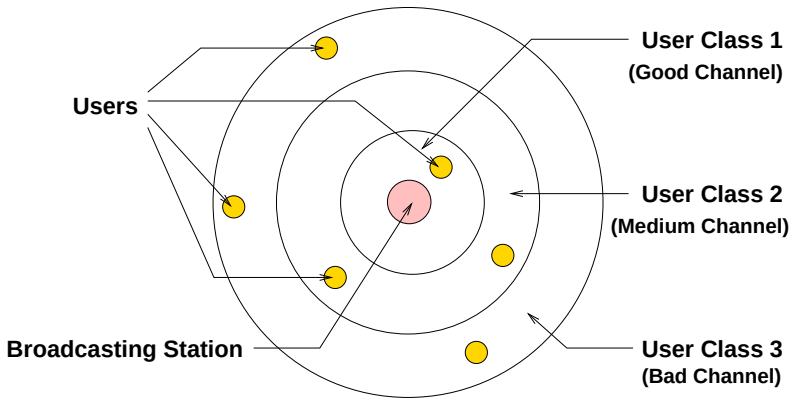

Figure 4.1 A simplified wireless broadcasting scenario.

successively refinable layers; each class of users can then receive a maximum number of layers within its ability and obtain a satisfactory performance. It has been shown that layered service indeed provides performance increase for all users [45], [46].

However, even with layered service and the performance increase it introduces, the problem of unequal performance for different users still exists, if somewhat alleviated. How to effectively and meaningfully evaluate the performance of the overall broadcasting system, given different classes of users, has been a major concern in the development of broadcasting schemes. Traditionally, people have developed heuristic and intuitive measures such as averaging the individual performance over all users, using weighted averaging schemes, or simply designing for the worst-case user. They either penalize certain class(es) of users, or are too ad-hoc to retain much theoretical significance.

We believe that a desirable and practical broadcast performance criterion should possess the following properties:

- **Fairness to all users:** No class of users should be penalized more than the other classes. For example, greater geographical distance from the broadcasting station is not a legitimate reason for heavier penalty in terms of performance.

- **Individual consideration:** Although no class of users should be unfairly treated, the fact that people do anticipate different service quality under different situations should also be taken into consideration (assuming the users are reasonably *informed*). For example, those users far away from the station would naturally expect the transmission quality to be somewhat inferior to that of the users next to the station. Thus, *our objective is*

to offer different classes of users the same level of relative satisfaction/disappointment, rather than exactly or approximately the same service quality.

- **Theoretical tractability:** We should be able to obtain the solution to the optimization problem based on the performance measure using established optimization methods, and the performance measure itself should have some theoretical significance.

4.2 Minimax Disappointment (MD)

To develop a good broadcast performance criterion based on the observations above, we first suggest a very general form of performance measure. Suppose we denote the raw video sequence as X, the receiver-reconstructed version for the ith user (class) as $\hat{X}_i$, and the total number of user (classes) as N, then a broadcast performance measure P can be expressed using the following equation:

$$P = \frac{1}{N} \sum_{i}^{N} w_i \mid f(X, \hat{X}_i) \mid^p \tag{4.1}$$

where $f(X, \hat{X}_i)$ is a predefined distance measure between X and $\hat{X}_i$, w_i is the *weighting coefficient* for the ith user (class), and p is the order of the norm operator. To optimize the broadcasting system, we set out to maximize performance measure P (or minimize it when P is defined as distortion). Here we list several commonly-seen special cases [21]:

1. **Average mean square error (MSE)**:

$$f(X, \hat{X}_i) = \frac{X - \hat{X}_i}{\sqrt{\mathcal{M}(X)}},\ w_i = 1,\ p = 2 \tag{4.2}$$

   where $\mathcal{M}(X)$ is the *cardinality* of X. This is probably the most commonly used criterion.

2. **Average mean absolute difference (MAD)**:

$$f(X, \hat{X}_i) = \frac{|X - \hat{X}_i|}{\mathcal{M}(X)},\ w_i = 1,\ p = 1 \tag{4.3}$$

   Often used in hardware implementation, when the square operation proves to be too expensive (for example, MPEG-2 block-matching uses MAD instead of MSE).

3. **Design for the worst user (in MSE sense)**:

$$f(X, \hat{X}_i) = \frac{(X - \hat{X}_i)^2}{\mathcal{M}(X)},\ w_i = 1,\ p = \infty \tag{4.4}$$

This approach provides *quality of service* (QoS) guarantee by designing the system toward the worst-case user ($p = \infty$ essentially yields the worst-case user's performance as the dominant term, $P = \max_i f(X, \hat{X}_i)$). Substituting $f(X, \hat{X})$ with other functions redefines service quality.

4. **Average peak signal-to-noise ratio (PSNR)**:

$$f(X, \hat{X}_i) = 10 \times \log_{10} \left(\frac{255 \times 255}{(X - \hat{X}_i)^2 / \mathcal{M}(X)} \right),\ w_i = 1,\ p = 1 \tag{4.5}$$

The log-scale PSNR criterion is commonly used in image/video processing because it matches to the human visual system (HVS) perception better than the MSE criterion.

5. **Phase correlation**:

$$f(X, \hat{X}_i) = \frac{\mathcal{F}(X) \times \mathcal{F}(\hat{X})}{|\mathcal{F}(X) \times \mathcal{F}(\hat{X})|},\ w_i = 1,\ p = 1 \tag{4.6}$$

where $\mathcal{F}(X)$ stands for the *Fourier transform* (FT) operation. The phase correlation criterion is used in image/video processing because it captures the visual similarity better than MSE or PSNR (for example, a darkened image can have a low PSNR from its original version, but the phase correlation would be high, indicating that it is similar to its original version).

If we use nonuniform values for w_i, then we essentially create *weighted* versions of the above criteria.

From a broadcast system's point of view, two commonly used performance criteria would be nonuniform w_i and $p \neq \infty$ (weighted averaging scheme), or uniform w_i and $p = \infty$ (design for worst user with QoS guarantee). Compared to these two common approaches, our proposed performance measure, named "*minimax disappointment*" (in analogy to the *minimax regret* concept in decision theory [47]), has the following advantages: (1) it does not employ any form of weighting, so the entire complex issue of designing weighting coefficients that are both theoretically meaningful and fair is avoided; (2) instead of sacrificing those users with good

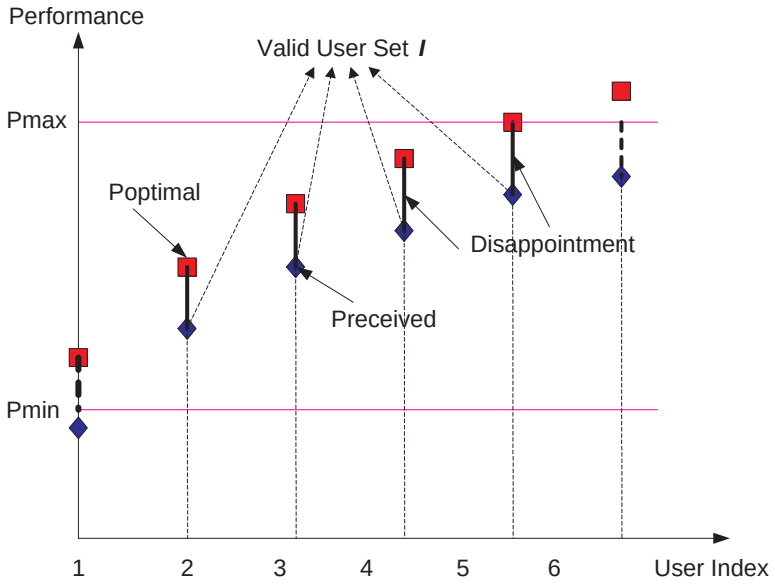

Figure 4.2 The minimax disappointment criterion.

channel conditions in order to guarantee a certain performance level for the worst-case users, MD trades off performance among all users and thus achieves approximately the same level of efficiency for all users. The disappointment criterion can be explained by Figure 4.2 and the following formula:

$$P = \max_{i \in I} (P_i - p_i) \tag{4.7}$$

$$I = \{ i \mid P_{min} \leq p_i \leq P_{max}, i \in 1, \ldots, N \} \tag{4.8}$$

where N is the number of user classes, P_i is the ith user's *expected best performance* (here *performance* is defined using any suitable metric; later we explain how this P_i is obtained using the JSCM principle), and p_i is its actually *received performance*. (Intuitively, for most users $P_i > p_i$ because we have to trade off performance between users. Equality can be achieved under certain circumstances but usually does not lead to the optimal configuration.) P_{min} and P_{max} are the upper and lower bounds for p_i; performances smaller than P_{min} are considered *unacceptable*, and performances greater than P_{max} are deemed *perfect*; those users whose performances satisfy these two inequality conditions form I, the *valid user set*.

We define $P_i - p_i$ as the ith user's *disappointment*, and our objective is to minimize the maximum performance degradation for all users in set I. In other words; all users in this set will be "disappointed" to a certain extent because they cannot receive their expected optimal

performance, and the maximum value of their "disappointment" P is the proposed performance measure: the smaller P is, the better the performance. We introduce the valid user set I to prevent users with extremely bad channels from having parasitic effects on the entire system, and avoid wasting resources on users whose received performances are already so good that any extra gain would not introduce perceptible effects. In this paper we make the assumption that the set I is predetermined before system optimization.

This performance measure does not penalize any class of users, and it is reasonable and fair because all users in set I know that everybody else is equally as satisfied/disappointed as they are. It is theoretically tractable because optimizing the performance simply falls into the minimax optimization framework.

The MD criterion can be extended to a minimax *relative* disappointment criterion (defined below):

$$P = \max_{i \in I} \left(\frac{P_i - p_i}{P_i} \right) \tag{4.9}$$

This criterion optionally scales each user's disappointment by its optimal expectation P_i. It reflects the fact that the user's *perceived* performance degradation is not proportional to the transmission system's absolute performance degradation: users who have extremely good performance might be willing to accept a relatively larger performance degradation since it will not significantly affect their perceived service quality; whereas users who already have mediocre or bad performance might be reluctant to accept any further performance degradation. Since this is simply a scaling operation, everything we describe for nonrelative minimax disappointment still applies.

The MD criterion can also be expressed as a special case of the general expression in Equation (4.1):

Minimax disappointment (MSE sense):

$$f(X, \hat{X}_i) = \frac{\mid (X - \bar{X}_i)^2 - (X - \hat{X}_i)^2 \mid}{\mathcal{M}(\mathcal{X})}, \; w_i = 1, \; p = \infty \tag{4.10}$$

$$\bar{X}_i = \arg\min_{\mathcal{C}} (X - \bar{X}_i)^2 \tag{4.11}$$

where $\mathcal{C}$ is the complete set of parameters to be optimized, such as the source and channel coding configurations in a JSCM system.

The existence of a unique solution for the maximization (or minimization) of performance measure P depends on the characteristics of the distance measure function $f(X, \hat{X}_i)$ and the order of norm operator, p.

1. $\mathbf{p = 2}$: The order-2 norm case is tractable because we can calculate the analytical expression of the first-order derivative of P and successfully apply gradient-descent-based algorithms. It is guaranteed to converge to a global minimum if the cost function is convex and has a finite number of discontinuities.

2. $\mathbf{p = 1}$: The order-1 norm case involves a change of sign in the zero point, for which the gradient is not defined. However, in the region where the cost function is monotonic, gradient-descent-based optimization methods can still be used.

3. $\mathbf{p = \infty}$: The infinite-norm case translates to a minimax optimization problem. It can be solved using the *first method of successive approximations* (a derivative of gradient-descent-based algorithm for solving minimax problems) when the cost function is convex, as we will show in later sections.

4. $\mathbf{p < \infty}$: For finite p-norm case, it is essentially similar to either case (1) or (2).

In our JSCM video transmission/broadcasting system, the distance measure function $f(X, \hat{X}_i)$ is determined by the rate-distortion function $R(D)$ of the source encoder, the performance curve $P_e(r)$ of the channel encoder, and the broadcast scheme. It is, under most practical situations, a continuous and convex function (see examples in Appendix A). Thus, the optimization of performance measure P in its most general form can usually be solved using gradient or gradient-derived algorithms. We concentrate on the MD measure as a meaningful special case.

4.3 Transaction Broadcasting Framework

To best implement layered service and JSCM broadcasting, we propose a broadcasting system (shown in Figure 4.3) based on a progressive video encoder and investigate several possible candidates. For channel protection we elect to use the RCPC channel encoder and, alternatively, a bit-power-adjustment approach. The encoding procedure can be generally described as follows:

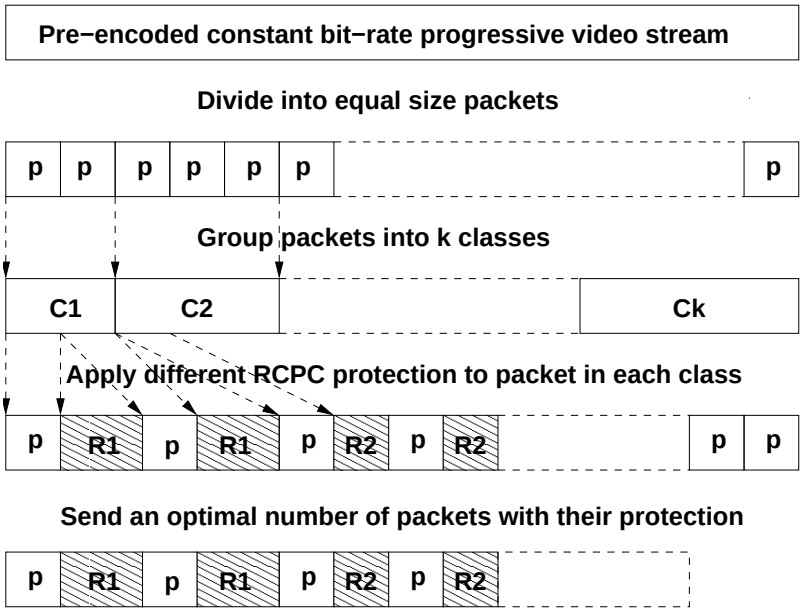

Figure 4.3 Proposed JSCM broadcasting framework.

1. **Pre-encoding:** The raw video frames are first pre-encoded using a fixed-rate progressive video encoder. Adjusting the encoding rate with a progressive video encoder is approximately equivalent to truncating the encoded bit-stream.

2. **Packetizing:** The encoded video stream is divided into equal-size video packets.

3. **Packet classification:** The packets are further categorized into k classes. The number of classes corresponds to the number of layers the broadcasting service provides.

4. **Channel encoding:** RCPC protection is applied to each packet in a class with a specific rate; it is possible for the least important (tail) packets to be transmitted without protection.

5. **Transmitting:** A certain number of RCPC-encoded packets are then transmitted. The number of transmitted packets is optimally determined by the JSCM procedure.

Information crucial to the decoding process, such as packet size, number of packets in each class, RCPC encoding rates, must be either predetermined and known to the receiver or pretransmitted to the receiver in the handshake or synchronization process. We do not investigate the implementation details of this operation in this thesis.

4.4 Simulation Results

In order to demonstrate the use of MD criterion and compare system performances, we choose to use the same set of simulation parameters for all four simulation systems:

1. Simulation sequence: fast-motion football test sequence cropped to the *quarter computer intermediate format* (QCIF).

2. Simulation channel: AWGN channels with SNRs ranging from 1 dB to 9 dB.

3. Number of user classes: 5 classes evenly spread out in the entire SNR range.

The last example is different from the previous three because it uses an energy-varying transmission with a power constraint instead of a rate constraint. We cannot directly compare its performance with the other three cases.

In order to adapt to the human visual system characteristics, we also elect to calculate disappointment using the log-scale PSNR. We use user-averaged-PSNR (UA-PSNR) as an alternative broadcast performance criterion to make general comparisons.

4.4.1 Motion-SPIHT and RCPC

To test the Motion-SPIHT-RCPC simulation system, the broadcasting station transmits the test sequence using the optimized system parameters with nine available RCPC protection rates. At the decoder end, a constant decode buffer size of approximately 1 bit/pixel is used. Simulation results are shown in Figure 4.4.

From Figure 4.4 we observe that the received performance for all users of our MD-optimized system is roughly the same distance from their individual expected best performance; in other words, they experienced the same level of disappointment, whereas in the other system configurations optimized toward one single user class, one or more users are invariably highly disappointed; i.e, their received performance is far below their best expectation. Thus, our system does give the optimal performance in terms of the MD criterion, achieving a maximum disappointment of 0.6703 dB.

4.4.2 3D-SPIHT and RCPC

In the case of 3D-SPIHT with RCPC simulation, the result is quite similar to the previous case. We again observe that systems designed for individual user class suffer from great

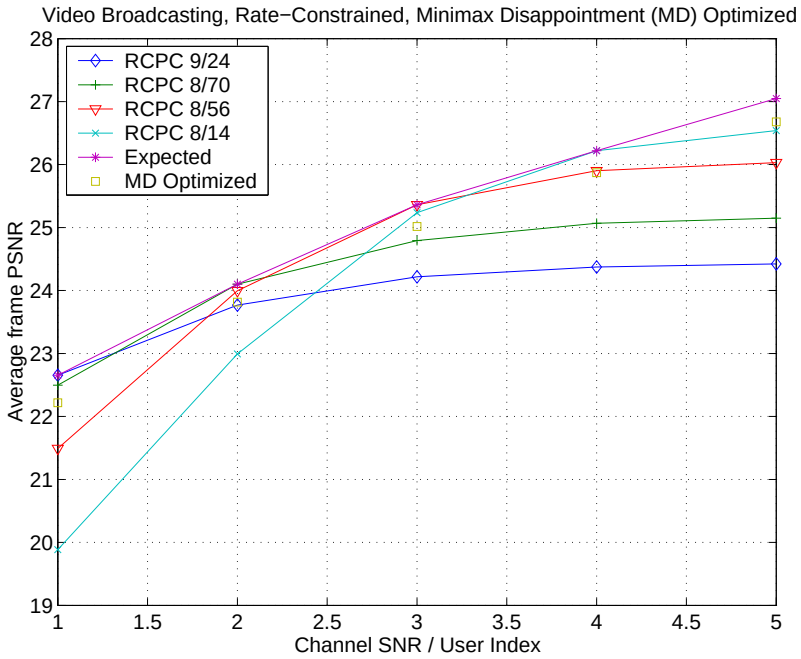

Figure 4.4 Simulation results for an MD-optimized video broadcasting system with five user classes in a rate-constrained situation, using Motion-SPIHT source coder and RCPC channel coder with nine available protection rates.

disappointment values for other users, and the MD-optimized system gives each user class approximately the same level of performance degradation (shown in Figure 4.5).

4.4.3 Layered-H.263 and RCPC

In the case of layered-H.263 with RCPC, we use the same simulation sequence and RCPC codes as before; the result is shown in Figure 4.6. In this case, we observe that not only the amount of disappointment experienced by each user varies, but also the level of maximum disappointment in the system is much greater (almost 2.4 dB) than the previous cases. This behavior is expected because of the inefficiency of retransmitting the coarse information in every layer. However, the better compression performance of this source coder may offset the larger disappointment in practical situations.

4.4.4 3D-SPIHT and bit-power

To demonstrate the generality of the MD criterion, we take an alternative approach by using the 3D-SPIHT-bit-power simulation system. We similarly make the assumption that we have

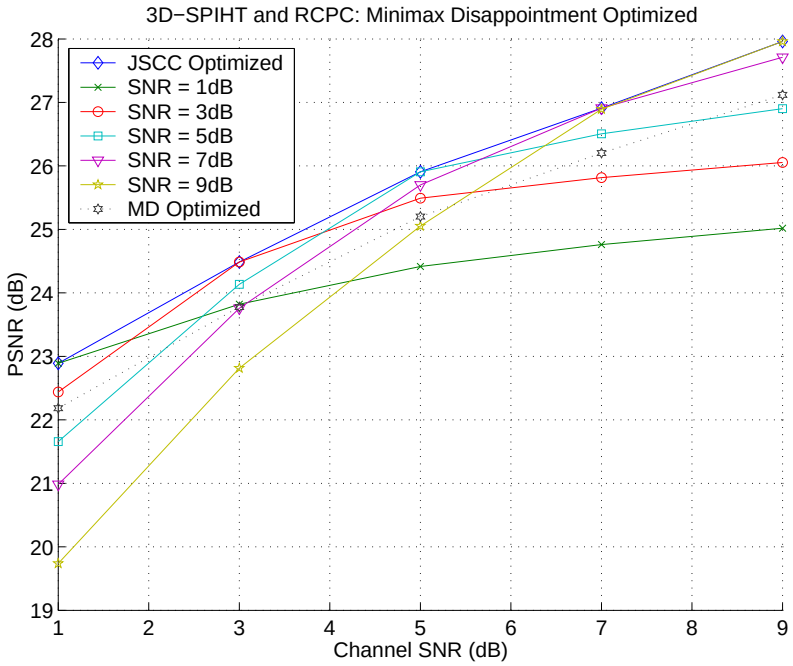

Figure 4.5 Simulation results for a MD-optimized video broadcasting system for five user classes in a channel-capacity-constrained situation, using 3D-SPIHT source coder and RCPC channel coder with nine available rates.

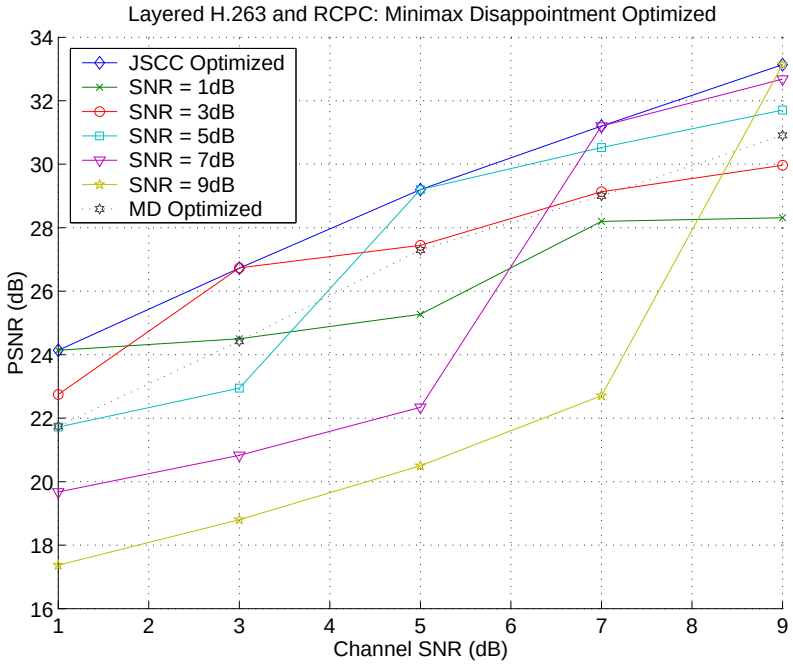

Figure 4.6 Simulation results for an MD-optimized video broadcasting system for five user classes in a channel-capacity-constrained situation, using layered-H.263 source coder and RCPC channel coder with nine available rates.

five classes of users mapped to different SNR levels of AWGN channels, and the transmission system can adjust the power for each bit. A total power constraint of E_{total} is also assumed. Simulation results are shown in Figure 4.7. Figure 4.8 shows the bit transmission power profiles for all the different system designs.

From the plot we again observe that all users experience approximately the same level of disappointment in our MD-optimized system, while using a transmission bit-energy profile optimized for any single user leads to one or more highly disappointed users. Our system achieves a maximum disappointment of 1.4418 dB.

From the plot we can also observe the performance curve for the case when we elect to use user-averaged PSNR as the performance criterion. In terms of user-averaged PSNR, the performance of the MD-optimized system is slightly inferior to that of the UA-PSNR-optimized system (by a mere 0.2 dB). However, in terms of maximum disappointment, the UA-PSNR-optimized system observes a maximum disappointment of 2.9871 dB, which is twice higher.

4.4.5 Result comparison

Comparing the result for Motion-SPIHT, 3D-SPIHT, and layered-H.263 under the capacity-constrained situation, we can see that for any individual user, using layered-H.263 with JSCM actually gives a higher absolute performance because of the H.263 encoder's much greater coding efficiency. However, the MD for layered-H.263 is much higher because its lack of progressive property makes unequal-error-protection difficult and inefficient. This example illustrates that the MD approach can be applied with success both to progressive and non-progressive video source coders, but that a highly-progressive coder is needed to maximally exploit the potential of MD broadcasting.

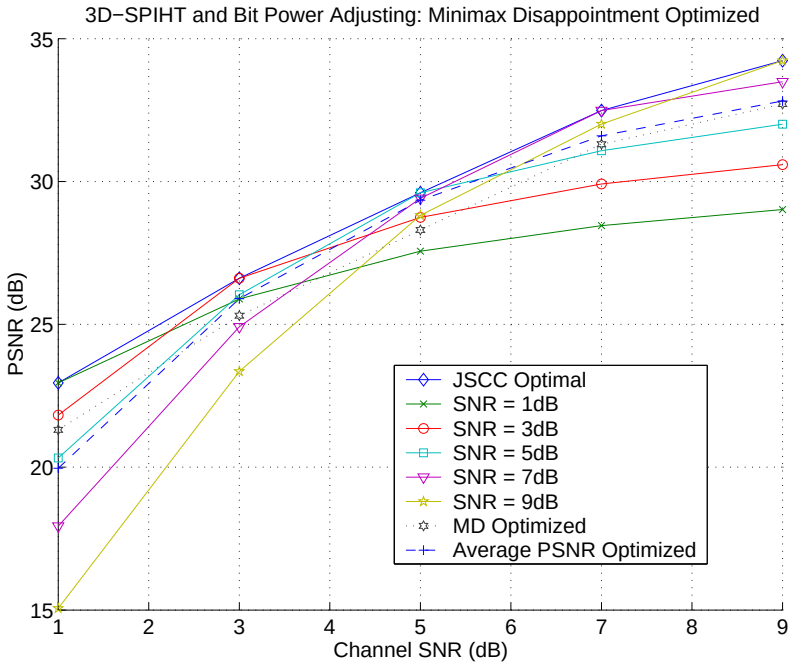

Figure 4.7 Simulation results for an MD-optimized video broadcasting system toward five user classes in a transmission-power-constrained situation, using 3D-SPIHT source coder and a bit-power-adjustable transmission scheme. User-averaged PSNR is used as an alternative performance criterion.

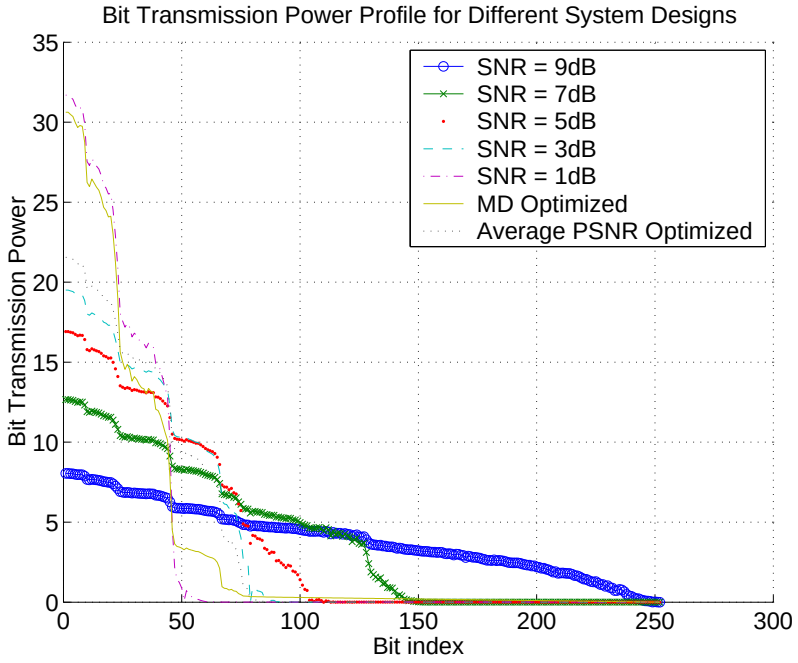

Figure 4.8 Bit transmission power profiles for different system designs.

CHAPTER 5

JOINT SOURCE-NETWORK MATCHING

Thanks to the explosion in network bandwidth and advancement in software development, the Internet has recently seen a flood of network video applications, ranging from videoconferencing packages such as Microsoft Netmeeting™ (where video sequences are captured, encoded and transmitted all *in real-time*) to video-streaming software such as Apple Quicktime™ and RealNetworks RealPlayer™ (where *preencoded* video is streamed from the server to a client upon request). Dot-com companies such as FilmSpeed and CinemaPop are experimenting with on-line movie distribution; education institutions such as the University of California at Berkeley have already established Web-based video-tutor programs. Network video applications are becoming increasingly popular.

In order to provide better support for video transmission over networks, two *conflicting* requirements need to be addressed [48]:

1. **Application requirements:** Video applications are delay-sensitive and require a semi-reliable channel. In addition, video streaming applications require isochronous processing and QoS guarantees from the end-to-end point of view, since stored video has an intrinsic transmission rate and demands a relatively constant bandwidth to deliver a stream with a certain quality.

2. **Network requirements:** Congestion is a frequent phenomenon in a network when the amount of ongoing traffic exceeds the total network link capacity [49]. End systems in the network usually utilize a congestion-control mechanism to determine the available bandwidth based on the state of the network. Thus, the available bandwidth could vary in an unpredictable and potentially wide-ranging fashion.

To satisfy these two requirements simultaneously, network video applications should be *bandwidth adaptive*: the application should attempt to deliver the video stream with the best

possible quality while the required bandwidth matches congestion-controlled rate-limit. In other words, the application should match the source coder and the network channel. Currently, most of the commercial network video applications do not perform source and network matching. They either transmit data at a near-constant rate that is negotiated during session initialization, or loosely adjust their transmission rates on long time-scales [48]. They are not responsive to network variations and therefore can neither take advantage of increased network bandwidth nor degrade gracefully during congestion (for example, during congestion, network video streaming through Microsoft Windows media player will stall and skip a large number of frames instead of showing continuous frames of somewhat inferior quality). Large-scale deployment of these video applications could result in severe interprotocol unfairness against well-behaved TCP-based traffic and possibly congestion collapse. Joint source-network matching is no longer merely a performance-enhancing technique, but a necessity to ensure a healthy network and end-user satisfaction.

Conceivably, the same joint source-channel matching principle demonstrated in the previous chapters can be applied to network video transmission with expected performance gains. However, the implementation of joint source-network-matched video transmission presents several unique challenges associated with the nature of network links. Various approaches have been proposed in the literature to solve those issues and to facilitate network video transmission: Freytes et al. [50] described *Kinesis*, an H.263+ video transmission system for IEEE 802.11 wireless networks; Cai and Chen [51] proposed a FEC-based robust video streaming system over a packet loss network with pre-interleaving; Vickers, et al. [52] defined a class of algorithms known as *source-adaptive multi-layered multicast* (SAMM) for real-time video distribution; Rejaie et al. [48] discussed layered quality adaptation for Internet video streaming; Kim et al. [53] developed a joint encoder and channel-rate control scheme and a dynamic bandwidth renegotiation method with traffic smoothing for *asynchronous transfer mode* (ATM) networks; Wu et al. [54] presented an end-to-end architecture for transporting MPEG-4 video over the Internet and an adaptive framework that specifically addresses video transport over wireless networks [46]; Kim et al. [55] demonstrated a TCP-friendly internet video streaming scheme using variable frame-rate encoding and interpolation; Liu et al. [56] investigated on-line dynamic bandwidth allocation algorithms for *variable-bit-rate* (VBR) video transmission; Zhu et al. [57] designed a new network-adaptive rate control scheme for scalable video communication in conjunction with unequal loss protection.

In this chapter, we discuss these various research challenges and propose our solution: the joint source-network matching protocol, which optimizes the end-to-end video transmission performance, using a TCP-friendly congestion-control algorithm and a built-in error-control mechanism.

5.1 JSNM Implementation Challenges

5.1.1 Dynamic network characteristics

The greatest difficulties in implementing JSNM video transmission arise from the dynamic and multi-user characteristics of network links. In a network video communication environment, there are, by definition, *multiple bidirectional concurrent* video sessions sharing the same network. Channel capacity with regard to any particular user in the network is a varying commodity instead of a fixed resource [58]. Packets with the same source and destination could travel entirely different paths with different delay parameters as a result of routing and queuing mechanisms.

Due to the sharing of network resources, the network link is not reserved for any particular user's exclusive use. Therefore, the previous "resource-greedy" approach employed by JSCM peer transmission systems is no longer applicable, since it will almost surely flood the network and cause instabilities. In addition, because the Internet does not currently manage utilization of its resources on a micro level, end systems are expected to be cooperative by reacting to network congestion properly and promptly; in other words, a *congestion-control* mechanism is required.

To perform congestion control, the following issues need to be addressed.

1. **Network modeling**

   One way to perform effective congestion-control is to assume an accurate underlying model for network traffic. Network modeling is an active research field: besides the conventional Poisson model and *multistate Markov chain* (MSMC) model [59], there are also the wavelet multifractal model [60], a mathematical model based on partial differential equations [61], a traffic model based on M-step Kalman filter [62], and many others.

   This thesis does not attempt to establish a new network model. Instead, it concentrates on comparing existing network models and choosing one appropriate for JSNM video transmission. Our simulations show that for our application, the MSMC is a good choice

with sufficient flexibility to describe network variations and desirable simplicity in terms of implementation.

2. **Network estimation**

   An alternative approach, which circumvents the necessity of choosing a network model versatile enough for all possible situations, estimates the network traffic situation in real-time based on receiver or router feedback. Real-time network estimation yields a model-independent, more accurate and responsive system if we can ensure the accuracy and timeliness of the feedback information.

   In our proposed JSNM protocol, the receiver generates a feedback packet containing important information regarding the network. The packet is delivered to the sender via a reliable protocol, and the sender can subsequently derive crucial network statistics from these data.

3. **TCP-friendly congestion control**

   The rapid deployment of multicast and real-time video streaming applications is highly likely to increase the percentage of non-TCP traffic in the Internet. These applications rarely perform congestion control in a TCP-friendly manner; in other words, they do not share the available bandwidth fairly with TCP-based applications such as Web browsers, *File Transfer Protocol* (FTP) clients, or *electronic mail* (Email) programs. The Internet community strongly fears that this trend could lead to starvation of TCP traffic or even congestion collapse [63] (i.e., the undesirable situation where the available bandwidth in a network is almost exclusively occupied by packets that are discarded because of congestion before they reach their destination). Thus, the mere existence of a congestion-control mechanism is not sufficient; it is also vital for our JSNM protocol to behave fairly with respect to coexistent TCP flows and degrade gracefully to TCP-like behavior under unfavorable conditions (i.e., conditions not designed for JSNM applications) [64]. A detailed survey on TCP-friendly congestion-control algorithms can be found in [65] and [66].

   In this thesis, we propose a TCP-friendly congestion-control mechanism roughly based on the original TCP congestion-control algorithm. It mimics the long-term behavior of TCP and yet does not have the saw-tooth pattern typical of TCP traffic, which introduces large variation in received video quality. The definition of TCP-friendliness and our proposed congestion-control algorithm are elaborated in Section 6.1 and Section 6.2.

4. **Routing and queuing**

   In a packet-switching network such as the Internet, packets usually travel through several intermediate nodes before they reach the final destination. Packets sent by the same transmitter to the same receiver could go through entirely different paths given the variation in network traffic or network topology, each path having its own characteristics such as transmission delay and queuing strategy. It is conceivable that we can ignore the IP layer details and regard the entire path as an entity, but since routers are essential parts of the network channel, a truly optimized JSNM system should also take them into consideration. Routers can be programmed to be JSNM-aware, so that *hints* regarding packet importance and delay requirement may be passed on to them and be used in the decision process. For example, in a channel-congestion situation, based on the hint information, the intermediate router may choose to promote packets with higher importance from the back of the queue to the front and transmit them first, or to drop those packets that are not expected to reach the receiver before their individual deadline.

   Research on JSNM routing support and queuing is out of the scope of this thesis. It is presented here to induce potential research interest.

5.1.2 Wireless network links

Networks, especially those with large-scale topology, are usually made of heterogeneous types of network links. For example, mobile video transmitters or receivers connect to their base stations via error-prone wireless links with high delay, while base stations are interconnected via high-speed large-capacity wireline links which are relatively error-free.

Networks involving wireless links are different from wireline-based networks in the following aspects:

1. **Less reliable**

   Wireless channels are typically much more noisy with fading phenomena and a significantly higher channel error rate, which puts a robustness requirement on video transmission systems.

2. **Higher fluctuations**

   The bandwidth for a user in a wireless mobile network could fluctuate for many reasons: (1) the mobile terminal moves across cell or network boundaries and hand-off occurs; (2)

there could be a burst of new traffic flows due to the introduction of new users; (3) the throughput may be reduced due to multipath fading, co-channel interference, or noise disturbances; and (4) the link capacity could vary with the changing distance between the base station and the mobile terminal. Consequently, bandwidth fluctuations pose a serious problem for real-time video if a wireless link is present.

Because of the popularity of wireless video applications, it is imperative that our proposed JSNM protocol recognizes the possible inclusion of wireless links in the end-to-end transmission path and address the above issues.

5.1.3 Need for new transport protocol

As mentioned previously, we consider none of the existing network protocols discussed in Section 2.5 entirely suitable for a joint source-network matching video transmission system because of their respective disadvantages: UDP does not provide congestion control and feedback mechanism, which are essential to JSNM; TCP employs retransmission techniques to guarantee reliability, which is sometimes not suitable for delay-sensitive applications such as video communication; even RTP, though flexible in the sense that it is deliberately incomplete, may prove to be an overkill for the task at hand; furthermore, it is not designed with wireless network links in mind. JSNM needs a new transport-layer protocol that provides congestion-control, a feedback mechanism, the ability to combat wireless fading, and avoids packet retransmission.

However, it is inefficient and unnecessary for us to develop an entirely new protocol from the ground up. We can take advantage of the services provided by existing protocols by modifying them or building a custom-designed layer on top. We design our proposed JSNM protocol on top of UDP to utilize its checksum and multiplexing service; the protocol follows the design principle of RTP and provides video session support. It features a congestion-control mechanism that is TCP-friendly, and includes two levels of protections to recover from both network congestion loss and wireless packet errors. For details, please refer to later sections.

5.1.4 Retransmission versus non-retransmission

The *automatic repeat request* (ARQ) mechanism exists in many protocols (such as TCP) that uses positive and negative acknowledgments with retransmission to ensure reliability. There is a debate whether or not ARQ techniques are suitable for real-time multimedia transmission such as videoconferencing. On one hand, ARQ ensures the reliable delivery of multimedia packets

that contain crucial information for decoding and avoids total decoding failure [67]; on the other hand, ARQ inevitably introduces unpredictable delays, which are detrimental to video communication, where only a small amount of delay can be tolerated.

At least two approaches can be used to solve this problem. One is to use a hybrid ARQ scheme [68] where only crucial packets evoke the ARQ mechanism; the other is to eliminate the necessity for ARQ by using the combination of progressive video source coders and an un-equal error protection scheme, where the successful transmission of crucial packets is ensured via the means of channel protection and error recovery instead of retransmission [57], [69]. Our investigation shows that the latter approach provides satisfactory results without relying on the relatively more complex and delay-sensitive retransmission-based schemes.

5.1.5 Network multicasting

Rather than sending a copy of each packet to each network user individually (unicast), multicasting sends the same packets simultaneously to all receivers. Though bandwidth-efficient, this delivery mechanism is undesirable because users are usually connected to the Internet at a heterogeneous set of rates [70].

Similar to wireless video broadcasting, JSNM can also benefit network video multicasting by applying MD performance criterion. One popular approach for video multicasting is receiver-driven layered multicast [70], [71], where video streams are encoded into several layers [72]; by subscribing to a subset, each user can have the best performance given the capacity limit of his network connection. JSNM and MD criterion can be used as a guideline to design these layers so that maximum end-to-end performance is achieved.

5.2 JSNM System Overview

In principle, our JSNM system is similar to the wireless peer transmission systems discussed in Chapter 3, since they simply replace the wireless channels with network links or packet-erasure channels. The same matching principle and optimization approach can be applied given an appropriate network channel model.

Figure 5.1 demonstrates the JSNM system framework, which is similar to the general JSCM system shown in Figure 3.2, pg. 24, with the following differences:

1. Instead of using a fixed channel capacity, JSNM contains a congestion-control unit that determines the current available bandwidth based on receiver feedback. JSNM's conges-

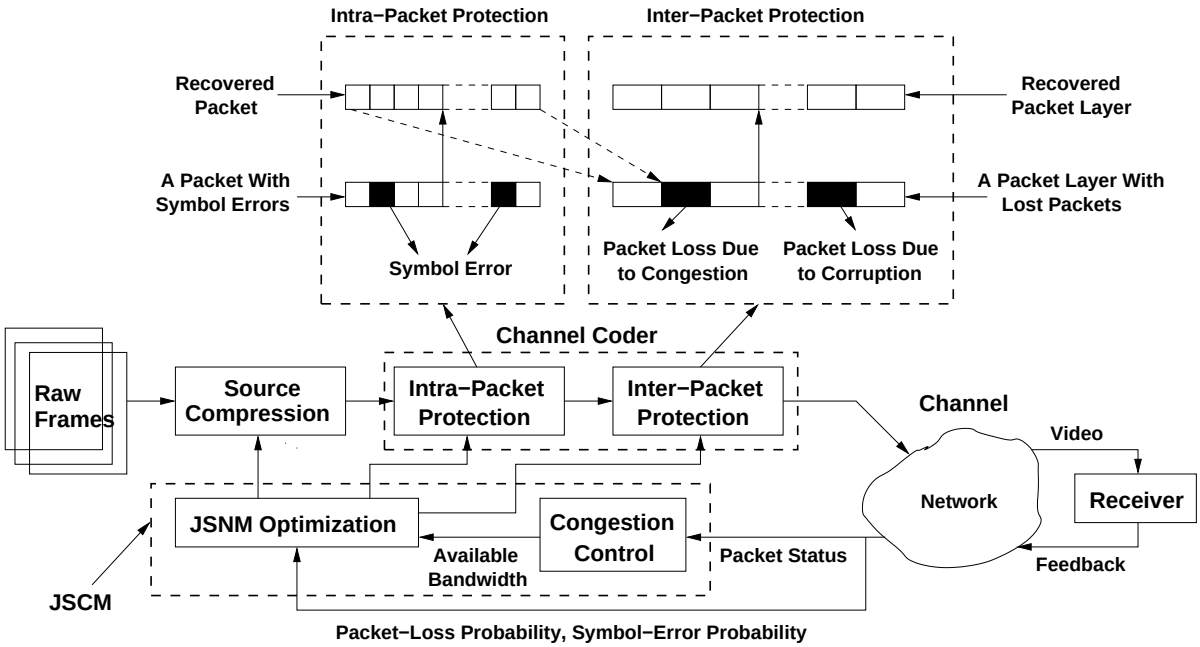

Figure 5.1 JSNM system overview.

tion control algorithm competes fairly with co-existing TCP data flows in the long term, and produces a smoother traffic pattern than that of TCP.

2. The point-to-point wireless channel is replaced with the network, which can include both wireless and wireline links.

3. The channel coder for JSNM consists of two concatenated operations: *intrapacket protection* and *interpacket protection* (see top portion of Figure 5.1). Intrapacket protection provides redundancy within a data packet to combat symbol errors caused by noisy wireless channels; interpacket protection recovers packets lost due to network congestion, time-out, or packet corruption due to transmission errors. JSNM is designed with wireless links in mind and recognizes the possible inclusion of noisy wireless links in the entire packet transmission path and takes corresponding measures to reduce its impact. In a pure wireline environment, intra-packet protection can be safely disabled.

In general, given the constraint on available channel capacity (estimated by the congestion-control algorithm), a JSNM protocol jointly optimizes the source compression rate and channel protection rate (both intrapacket protection and interpacket protection) to achieve the best end-to-end video transmission performance. Under good channel conditions, it decreases the amount of protection to reduce distortion caused by source compression; under bad channel

conditions it applies greater compression to the video source in exchange for more bits to spend on channel protection. JSNM ensures the *expected* optimal performance by trading off source and channel rates.

5.3 JSNM System Implementation

In this section we introduce the JSNM implementation based on our JSNM protocol. The details of the protocol itself can be found in Chapter 6 and Appendix E; here we discuss the two most important components in our implementation: the video sender and receiver.

5.3.1 The JSNM video sender

Our proposed general JSNM system diagram (the sender side) is shown in Figure 5.2. The encoding and transmitting procedure can be briefly described as follows (the session state machine is shown in Figure 5.3):

1. **Session establishing:** The sender sends (via TCP) a *request* (REQ) message to the receiver and waits for the *acknowledgment* (ACK) message from the receiver. The REQ message also carries crucial decoding information such as video type (color or grayscale), video frame size, GOP size (number of frames in each GOP), target bit-rate, the initial sequence number and timestamp, etc. If the receiver accepts this request, it sends back an ACK message and waits for the sender to initialize the session; if not, it sends a *finish* (FIN) message to the server to indicate rejection of the request. The ACK message also contains useful information, such as the receiver's current timestamp and an estimate of the current network condition, to help the sender decide on the initial set of parameters to use. The sender finishes the hand-shaking process by sending an *initialization* (INI) message to the receiver. The INI message contains additional information that could be useful to the receiver. The video transmission session is now established (for details of the information carried in each message, please refer to Appendix E).

   (In this thesis, we make the assumption that the handshaking messages are always correctly received in time by the receiver.)

2. **Video GOP encoding:** The raw video frame sequence is first divided into GOPs of size M. Each GOP is then fed into a *progressive-fine-granularity scalable* (PFGS) video

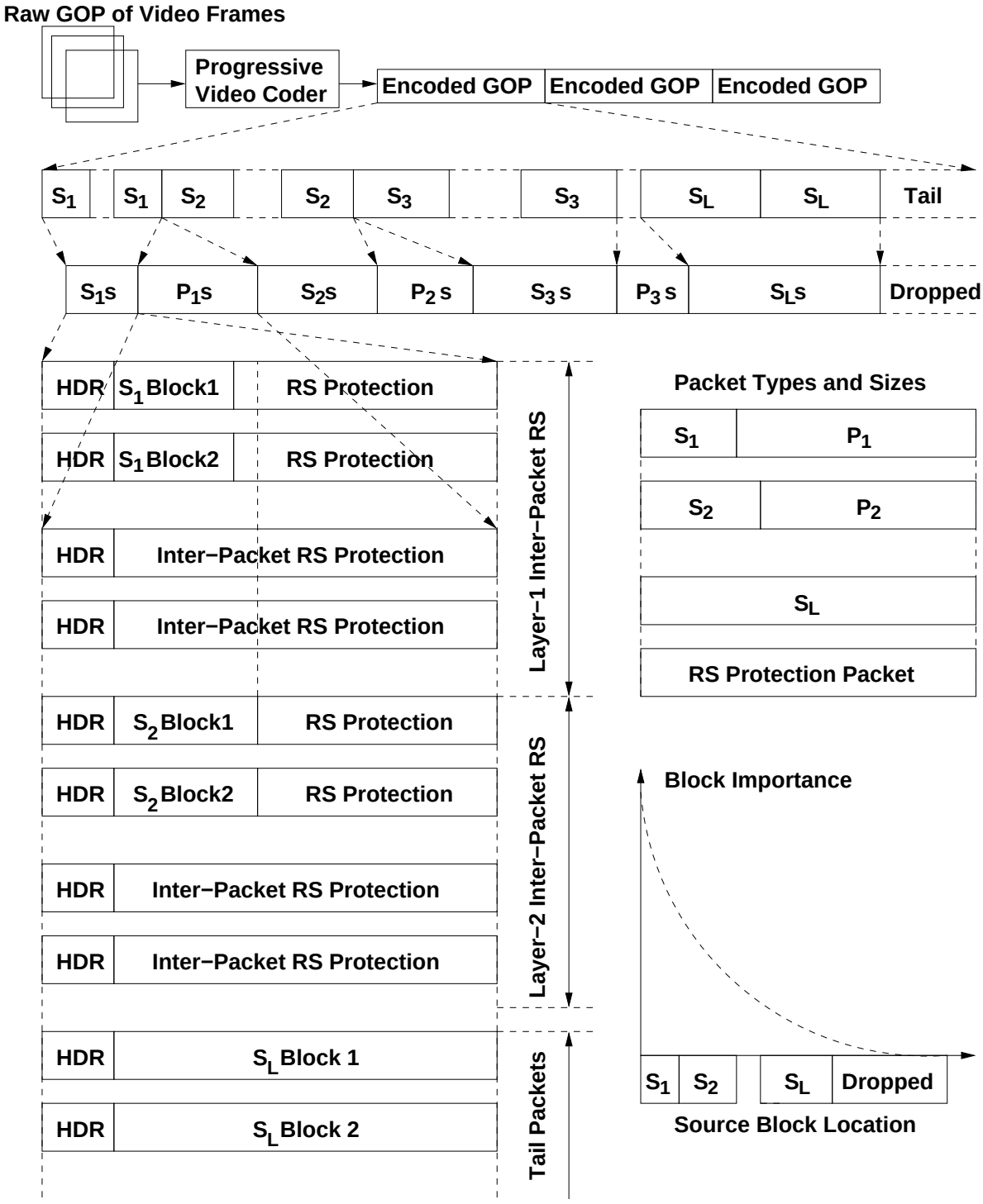

Figure 5.2 JSNM sender diagram.

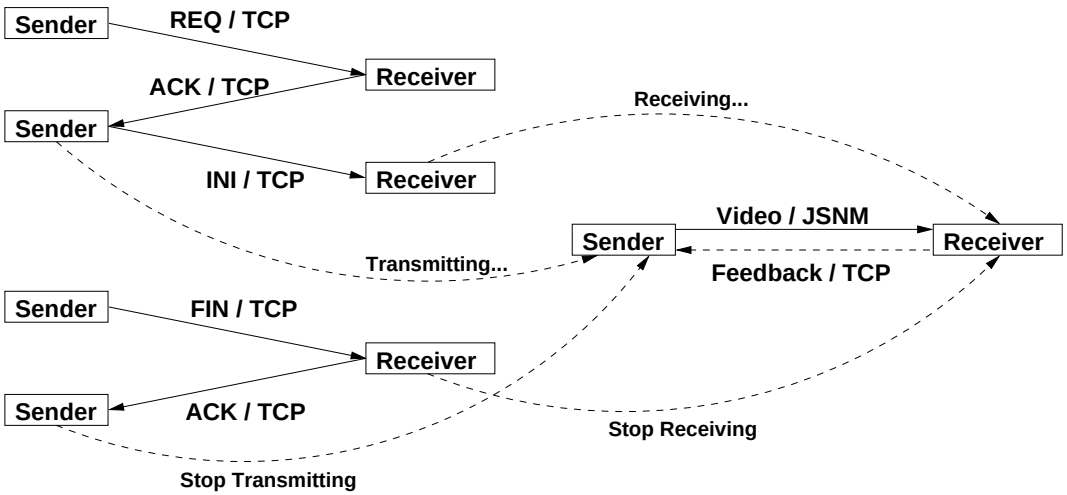

Figure 5.3 JSNM system state machine.

encoder (for example, 3D-SPIHT, which we use in our simulations), which outputs consecutively encoded GOP streams of approximately the same length (the encoding rate specification might not be exact).

3. **Stream segmentation:** Each GOP stream is further divided into L *layers* of blocks (shown in Figure 5.2 as S_1 to S_L). The first $L-1$ layers symbolize blocks with different importance (corresponding to $L-1$ different receiver performance levels), and the last (L-th) layer, contains trivial tail blocks that can be safely dropped without significant impact on performance. Blocks belonging to the first $L-1$ layers are packetized and transmitted; blocks belonging to the last layer can be dropped without being sent. The size of blocks in each layer $K_i - 1$, the number of blocks in each layer k_i, and the number of packets in each layer n_i, are all determined by the JSNM optimization procedure (described in Section 6.3.2) based on receiver feedback.

4. **Packetization:** This step packetizes all the blocks in three stages:

   a. **Error detection checksum:** A checksum byte is appended to all video source information blocks as a final check for the correctness of received information.

   b. **Intrapacket protection:** The block, together with the checksum byte, is encoded using a *systematic* RS encoder (shown in a Figure 5.2 as RS encoding in the horizontal direction). A *packet header* (for packet header syntax, refer to Appendix E.1) is then prefixed to the encoded block. The intrapacket RS encoder has the same codeword

length N for all layers but different encoding rates K_i for each layer. *Intrapacket RS protection is used to combat packet transmission errors due to wireless fading.*

c. **Interpacket protection:** After all the blocks from a certain layer are packetized, we generate protection packets for this layer by performing *systematic* RS encoding *across* the packets (shown in Figure (5.2) as RS encoding in the vertical direction). Both the interpacket RS codeword length n_i and encoding rate k_i could vary for different layers. *Interpacket RS protection is used to combat packet loss due to channel congestion, timeout, and other reasons.*

d. **Tail-packets:** Neither intrapacket protection nor interpacket protection is applied to blocks from the Lth layer. They are sent "as is" (with the header prefix) using a best-effort delivery strategy.

5. **Video delivery:** All the packets for this GOP are then transmitted at a constant rate within the scheduled delivery time. The transmitter goes back to step 1 to process the next GOP.

6. **Session termination:** All video GOPs have been transmitted. Sender sends the FIN message to the receiver and waits for the receiver to acknowledge. Once acknowledged, the transmitter ends the current video transmission session.

To summarize, three different types of packets are generated and transmitted in the entire transmitting process: the *protected source packets*, which contain video source information and intrapacket RS protection; the *RS protection packets*, which contain interpacket RS protection information; the *unprotected source packets*, which contains only video source information and no intrapacket or interpacket RS protection.

Some important observations about the JSNM transmitter scheme follow:

1. Corresponding to their relative location in the progressive video bitstream, the importance of blocks from different layers is different. An ith layer block (packet) is more important than an $(i+1)$th layer block (packet). *Packets with smaller layer index need greater protection.*

2. *The intrapacket RS encoding rate (K_i/N) increases with i.* This is not a physical constraint but a logical conclusion from their relative importance stated above, if we want to achieve the optimal end-to-end performance. As a direct result, the source information

block length $K_i - 1$ stays the same within the same layer but increases with i; i.e, an ith layer block is shorter than an $(i+1)$th layer block. The intrapacket protection for an ith layer packet T_i is greater than that of an $(i+1)$th layer packet T_{i+1} (since the intrapacket RS codeword length N stays the same across layers, and $N = K_i + 2T_i = K_{i+1} + 2T_{i+1}$).

3. *The interpacket RS encoding rate (k_i/n_i) generally increases with i.* In other words, the interpacket RS protection rate is decreasing with i, and the *combined effect* of the intrapacket and interpacket protection decreases with i.

4. The number of packets in each layer n_i, the intra-packet RS encoding rate K_i/N, and the interpacket RS encoding rate k_i/n_i, are determined by the JSNM algorithm (discussed in Section 6.3.2). The JSNM algorithm relies on *periodic reliable receiver information feedback* to make corresponding parameter adjustments.

5. The packet length N and the size (in packets) of each intra-packet protection layer n_i must be *valid RS codeword length*. For simplicity, we fix N with a value of 255 in our systems. n_i could be all the valid values ($2^m - 1$, m is an integer) up to 31. Similarly, both K_i and k_i have to be odd integers smaller than N and n_i, respectively.

6. Both the intrapacket RS coding configuration (N, K_i, T_i) and the interpacket RS coding configuration (n_i, k_i, t_i) *must be known* at the receiver to ensure correct RS decoding. The most obvious solution (namely, pretransmitting those information to the receiver with heavy protection or through a reliable protocol like TCP) is inefficient and against the design philosophy of JSNMP. Our proposed solution is to embed the RS coding configurations in the header section of every packet; thus, they are inherently protected by large redundancy, which enables the receiver to make a well-informed guess of the RS coding parameters. For details of the receiver RS coding parameter recovery procedure, please refer to Appendix E.2.

7. Our proposed JSNM protocol is a transport protocol on top of UDP. Every JSNM packet is embedded in a UDP packet with its own header and checksum (which could be used to check for transmission errors). It provides a clean interface, or set of *application programming interfaces* (APIs), to the upper-layer applications.

5.3.2 The JSNM video receiver

Unlike the receivers in other JSCM systems, the JSNM video receiver is fairly complex because it not only performs source-channel decoding of the received information, but also carries out two additional important tasks: determining the correct RS decoding parameters, constructing and sending the feedback information packet. The receiver's tasks can be described as follows:

1. **Session establishing:** After receiving the REQ message from the sender, the receiver acknowledges by sending an ACK message (or a FIN message if it decides to reject the request), which also carries important information such as the receiver's estimated bandwidth to facilitate the sender in choosing the initial set of coding parameters to use. The receiver then waits for the INI message from the sender. Once the INI message arrives, the receiver collects all the useful information and waits for the first packet to arrive.

2. **Packet header recovery:** Assuming there is a wireless link in the entire network channel, packet headers, which contain crucial RS decoding information and sequence numbers, could have been corrupted by transmission errors. The receiver identifies the type of each packet, recovers the sequence number, extracts the intra-inter RS encoding parameters from the packet headers, and determines the layer boundaries. If the receiver fails to recover the header, the packet is labeled as corrupted.

3. **Interpacket RS decoding:** For the ith layer, if the receiver has received more than k_i packets (regardless of packet type), then it can recover all the packets via RS erasure decoding (assuming we know the locations of erased packets from the sequence numbers). If the receiver has received less than k_i packets, it can recover the first consecutive set of type-1 packets because we use a systematic RS encoder.

4. **Intrapacket RS decoding:** For each type-1 packet, the receiver performs RS decoding within the packet and attempts to correct all transmission errors. If the decoded block has a checksum mismatch, then RS decoding has failed and the receiver labels the packet as corrupted. The receiver also records the number of symbol errors observed.

5. **Video bit-stream reassembly:** The video source information portions of each type-1 packet are ordered by the packet sequence number and concatenated to form the original progressive video bit-stream. For the regions corresponding to lost packets, zeroes are

filled; for the regions corresponding to corrupted packets, decoded blocks are filled in "as-is" in the hope that they would increase the decoding quality.

6. **Sending feedback packet:** The receiver constructs a feedback packet and transmits it back to the sender via a reliable protocol such as TCP. The feedback packet contains important channel error-rate related information, plus an array of bytes with every two-bit allocated to reflect the four possible states of a packet (See Appendix E.3 for details). The four possible states are: "correct" (received in time and correctly decoded), "corrupted" (received in time but unable to decode), "lost" (never received or received after deadline), and "unknown" (which corresponds to any unreceived type-3 packets that are either lost or not sent out at all). The sender can then use this information to perform JSCM.

7. **Session termination:** The receiver either voluntarily terminates the current video session or waits for the sender's FIN message. Upon receiving the FIN message, the receiver plays back the last GOP and terminates the video session.

A few important details of the receiver scheme are:

1. The feedback information, sent via a TCP packet or any reliable method, involves a delay, which means it might not have arrived when the sender needs it for the optimization of transmitting the next GOP. In the case when the feedback packet fails to reach the sender in time, the sender will use the same coding and transmitting parameters for the previous GOP in the hope that they are still valid. The sender stores a feedback packet whenever it arrives, and uses its content to optimize the transmission of the next GOP.

2. Another important issue with feedback information is its accuracy. In our simulation, the information-feedback packet is sent at fixed intervals. When the packet reaches the sender, the information it contains about the channel, such as the packet loss ratio, might already be out of date and thus inaccurate. In this thesis, we make two assumptions to ensure the accuracy of the feedback information: (1) we assume that the receiver sends the feedback frequently enough that the current network traffic status is truthfully represented (in our simulation the receiver sends feedback for every GOP it receives) and (2) we assume that we encounter only long-term fading in any possible wireless links, with a fading duration much longer than the feedback update period, and thus the feedback information about wireless channel error rate is still valid when the sender receive it. Occasional fast fading errors can be corrected by the extra *safe-guard* protection discussed in Section 6.3.2.

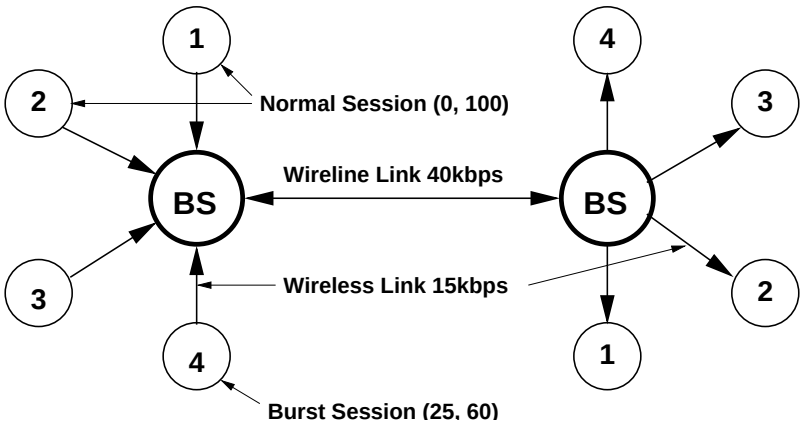

Figure 5.4 Network simulation topology.

5.4 Simulation Results

The real-life testing of the functionality of a complete network protocol is a complex task that requires considerable resources. Hence, we verify the performance of our JSNM protocol mostly based on simulation results from the freely available network simulation software package *NS*-2 from the University of California at Berkeley.

5.4.1 Network simulation topology

We use the typical "dumb-bell" network topology shown in Figure 5.4. The network contains four individual video sender-receiver pairs and two base-stations. The four video sessions start at different time instants. Each end-user is connected to the base station using a wireless link of capacity 15 kb/s and a delay of 100 ms, which simulates a mobile phone link with a good channel condition. The two base-stations are connected using a wireline link of capacity 40 kb/s (it is deliberately set to be smaller than the sum of the capacity of four individual wireless links to create network congestion when there are four concurrent video sessions) with a delay of 10 ms. The four video sessions start and terminate at different time instants to simulate the dynamic nature of network channels and traffic bursts. Video Sessions 1, 2, and 3 all start transmitting video at time instant 0 s and terminate video session at time $t = 100$ s; Video Session 4 enters the system at time $t = 25$ s, causes network traffic congestion, and terminates video session at time $t = 60$ s.

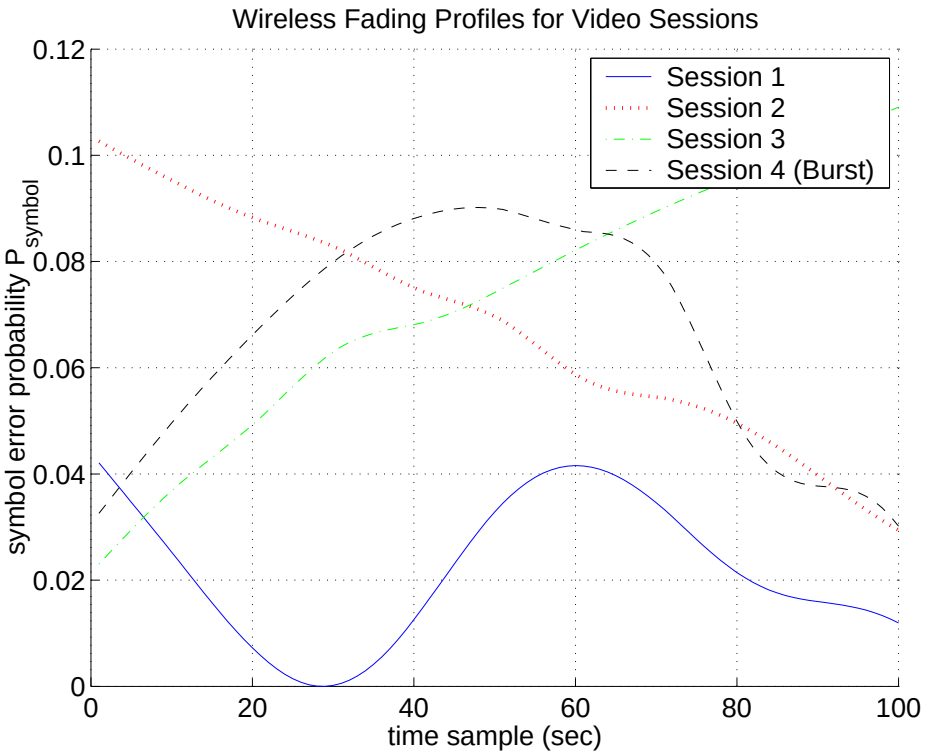

Figure 5.5 Wireless channel fading profiles.

5.4.2 Simulated wireless fading channels

We simulate the wireless mobile phone links with four fading profiles (generated off-line and shown in Figure 5.5) for the four video sessions. Video Session 1 observes a relatively good channel with slight SNR variations; Video Session 2 observes a constantly improving channel; Video Session 3 observes a constantly degrading channel; and Video Session 4 observes the peak of a relatively bad fading channel.

5.4.3 Simulation parameters

We use the JSNM system described in Section 5.3. The system uses 3D-SPIHT as the progressive video encoder, RS codes as the FEC channel encoder, and the JSNM protocol for the transport layer. The sender performs TCP-friendly congestion-control and packet loss-error control for every GOP, the receiver periodically sends a feedback packet via TCP. The standard "Miss America" sequence is used at QCIF resolution with a GOP size of 16 and total sequence length of 1600 frames. The sender transmits video at a constant rate of 1 GOP per second (16 *frames per second* (fps), which is normally used in videoconferencing) and the receiver plays

back the sequence at the same rate. Unlimited buffer size is assumed for both the sender and the receiver. The two base stations employ the "drop-tail" strategy for queued packets during network congestion.

The following section shows a comparison of the performance between our JSNM system and those of systems using other non-JSNM protocols. We also investigate the performance of our JSNM system in a pure-wireline environment with coexisting flows using the TCP. To correctly interpret the results, we again note that each user has a distinct channel fading profile as shown in Figure 5.5.

5.4.4 Results and discussions

Each figure in this section shows the performance of a particular transmission system, based on 100 s of simulation. Each curve shows the received GOP's PSNR trace for a particular video session. From the figures we can observe the following:

We conclude that our system behaves the way that we expect. (It is difficult to compare our system with schemes proposed by other papers, since most of them assume a wireless channel capacity of at least 60 kb/s, while ours assumes a more realistic 15 kb/s per channel.). A comparison between our system and non-JSNM systems reveals the following:

1. Figure 5.6 shows the performance when we use UDP as the transport protocol, which provides neither congestion control nor error control. The PSNR traces, although relatively smooth because the sending rate is constant, are actually quite bad. First, because there is no congestion control, when Video Session 4 enters the system, the performance for all sessions dropped to as low as 13 dB, about 10 dB lower than the JSNM system in average. Second, due to the lack of error control, video PSNR is low when the wireless channel experiences a high probability of error, in Session 2 in $time = 0 - 20$ s and Session 3 in $time = 60 - 100$ s. Third, even when the traffic burst is not present and the channel is relatively good, video PSNR performance is still slightly lower than the JSNM system by approximately 1 dB.

2. Figure 5.7 shows the performance when we use TCP as the transport protocol. Now there is congestion control, but no error control. We observe that because of the existence of congestion control, the impact of the burst user is significantly reduced. However, session PSNR performance is inferior when the channel error probability is high because of the

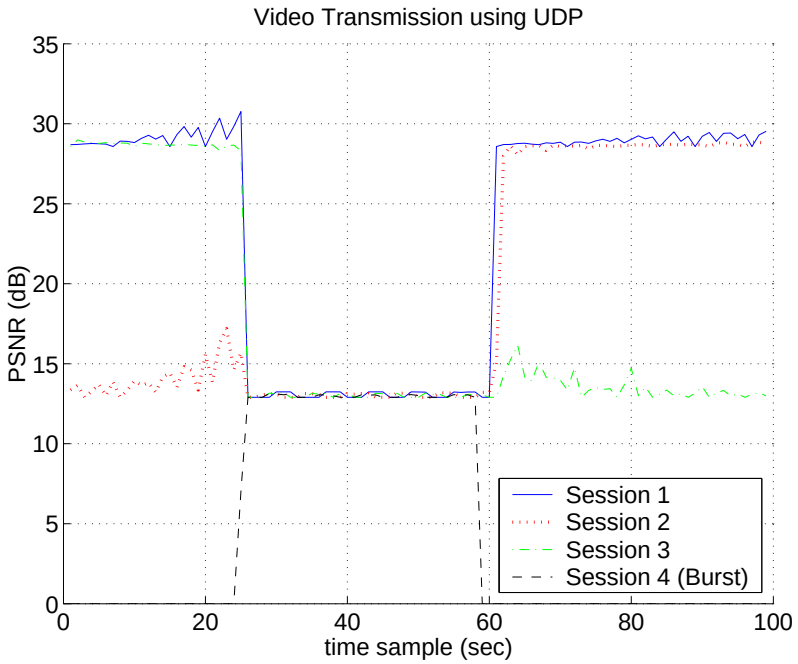

Figure 5.6 Simulation results for a UDP-based network video-transmission system with four concurrent video sessions and using 3D-SPIHT video source encoder and RS codes as channel encoder.

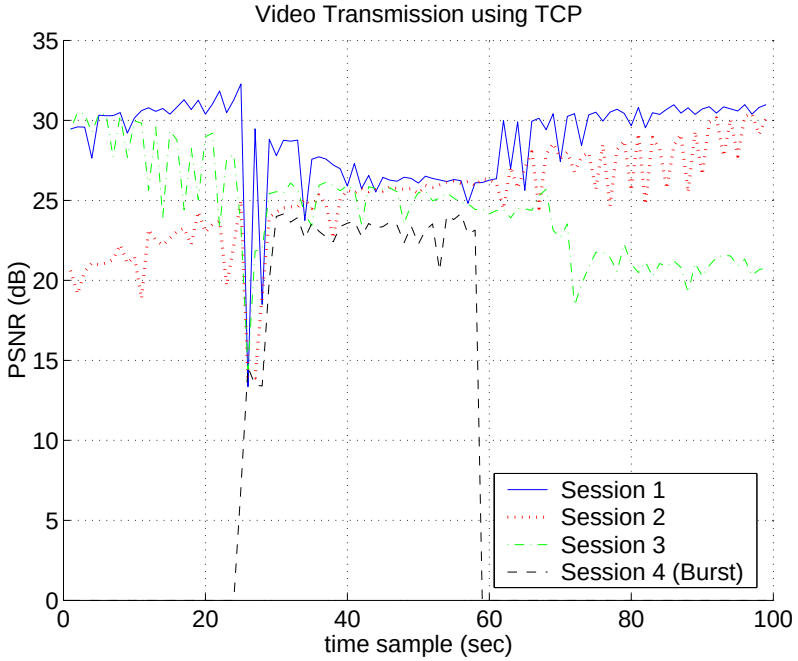

Figure 5.7 Simulation results for a TCP-based network video-transmission system with four concurrent video sessions and using 3D-SPIHT video source encoder and RS codes as channel encoder.

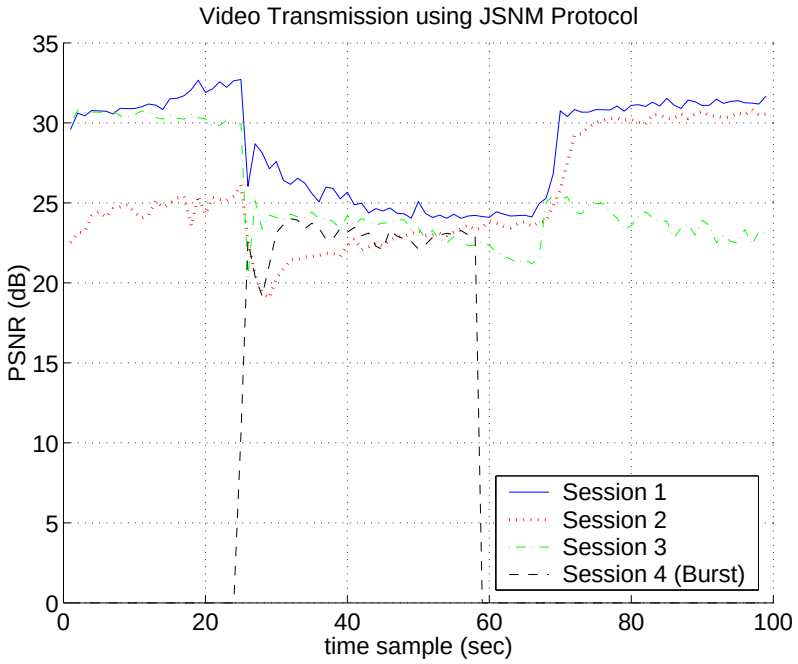

Figure 5.8 Simulation results for a JSNMP-based network video-transmission system with four concurrent video sessions and using 3D-SPIHT video source encoder and RS codes as channel encoder.

lack of error control. The high amount of delay introduced by TCP's retransmission mechanism also slightly reduces overall performance. TCP's "saw-tooth" like traffic behavior contributes to the higher variation of the performance curves.

3. Figure 5.8 shows the performance of our JSNMP-based system. We can observe that due to the congestion-control mechanism, when the burst traffic occurs, the performance of all video sessions degrade gracefully without drastic decrease as in the UDP-based system; second, the performance curves are much smoother than those of the TCP-based system; third, because of the error control mechanism, the impact of packet errors caused by wireless fading is reduced, and the performance curves under a bad channel are higher than those of the UDP and TCP-based systems.

4. To demonstrate the necessity of having intrapacket RS protection, we disabled the intrapacket RS protection mechanism in our JSNM-based system in our simulation. The result (Figure 5.9) shows that with intrapacket protection disabled, due to packet corruption caused by transmission errors and the lack of a retransmission scheme, performance for the JSNMP-based system in some cases could drop to the level of that of the UDP-based

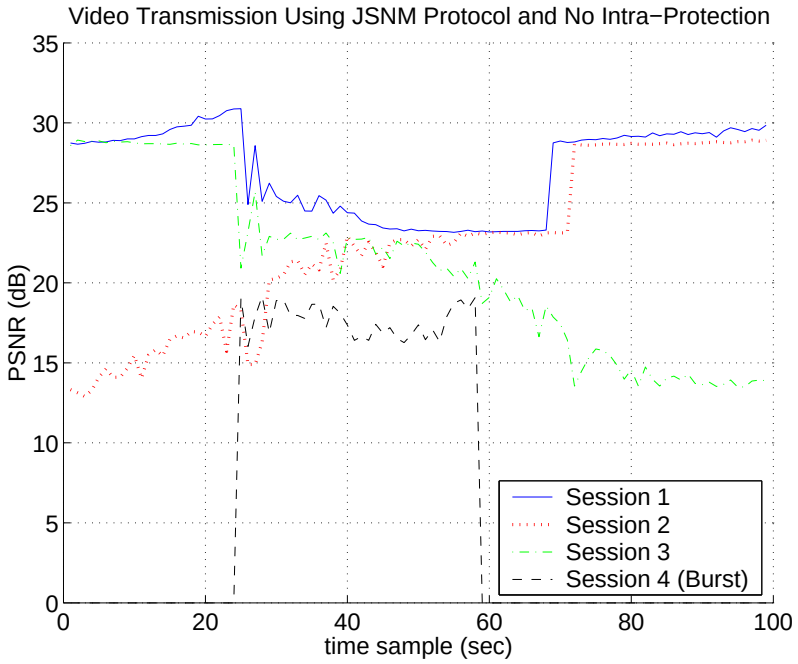

Figure 5.9 Simulation results for a JSNMP-based network video-transmission system with four concurrent video sessions and using 3D-SPIHT video source encoder and RS codes as channel encoder. Intrapacket RS protection is disabled.

system, much lower than those of the TCP-based and JSNMP-based (with intrapacket protection enabled) systems. *Having symbol-level error-correction capability can drastically improve performance for wireless mobile video transmission systems.*

5. Although JSNMP is designed with wireless mobile video transmission in mind, its performance in a pure wireline environment is also important and may be of higher value in practice, since most wireless network infrastructure has yet to support symbol-level manipulation ability to take advantage of JSNMP's intrapacket RS protection mechanism. It is also of interest to investigate whether or not JSNMP flows can coexist with TCP flows in a fair fashion.

   Figure 5.10 shows the performance of JSNMP in the same simulation system but with all wireless links replaced by wireline links and Video Session 1 using TCP instead of JSNMP as the transport protocol. We observe that without the effect of fading, performance for all video sessions have greatly increased, and the effect of network congestion is well controlled by JSNMP's congestion-control algorithm. Due to JSNMP's smoother traffic regulation, its performance is better than that of TCP at the beginning stage of network

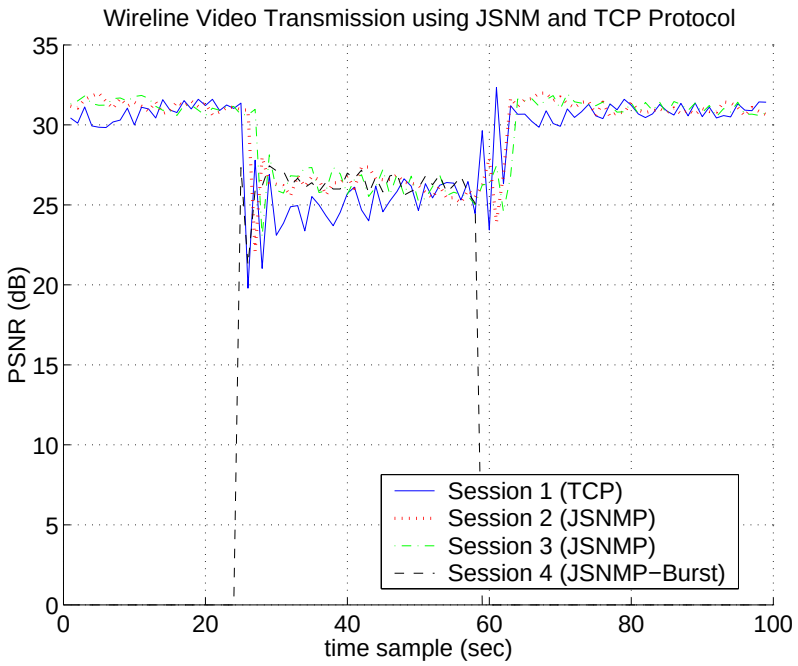

Figure 5.10 Simulation results for a JSNMP-based network video-transmission system with four concurrent video sessions and using 3D-SPIHT video source encoder and RS codes as channel encoder. All links are wireline links. Video Session 1 uses TCP instead of JSNMP as the transport protocol.

congestion, but generally converges to TCP performance when traffic stablizes. Our claim that JSNMP congestion control is TCP-friendly in the long-term behavior is again demonstrated.

For details of the JSNM protocol such as packet format, the congestion control and error control algorithm, please refer to Chapter 6. Here, in conclusion, we list a few of the observations we made regarding the simulation results:

1. The JSCM principle can be applied to network video communication systems with significant performance improvements compared to conventional systems using standard non-JSNM protocols.

2. Simulation shows that intrapacket protection is required for the JSNM system to have good performance.

3. The JSNM protocol performs equally well in a pure wireline environment with coexisting TCP flows. It is truly TCP-friendly in its long-term behavior. It also produces smoother quality variations than TCP.

4. Simulation shows that using a scalable video encoder in conjunction with unequal error protection techniques is an acceptable alternative to ARQ-based systems. It also has the advantage of requiring less transmission overhead and functions better for delay-sensitive application. A hybrid ARQ-FEC scheme is certainly an interesting further research possibility.

CHAPTER 6

A JSNM TRANSPORT PROTOCOL

To successfully implement the JSNM video transmission system introduced in Section 5.3 and achieve the optimal end-to-end video transmission performance, we need to answer the following questions:

1. How many packets should be sent for the current GOP?

2. What are the interpacket RS coding parameters (the values for all n_i, k_i and t_i, where i is the layer index)?

3. What are the intrapacket RS coding parameters for all type-1 packets (the values for all N_i, K_i and T_i)?

Obtaining the answers to these three questions correspond to the three major tasks that the JSNM protocol needs to perform: effective congestion control, recovery of packet loss due to network congestion and packet corruption, and correction of corrupted packets caused by wireless transmission errors. We denote them as network congestion control, packet loss control, and packet error control, respectively.

In the most complete sense of JSNM, these three aspects should be jointly optimized for best end-to-end performance. However, we recognize the fact that the Internet is a *distributed* system without a centralized mechanism to regulate network traffic; therefore, users must cooperate with each other in determining available bandwidth without being resource greedy. It is necessary that we decouple the congestion-control mechanism from the JSNM optimization. Thus, the JSNM protocol involves two indepedent steps: first the sending rate is determined based on congestion control, then the amount of protection for packet loss and error control is determined using JSCM.

In this chapter we propose a joint source-network matching transport protocol, which includes a TCP-friendly congestion-control algorithm and a packet loss-error control algorithm.

For implementation-level details such as packet format and header specification, please refer to Appendix E.

6.1 TCP-Friendly Congestion Control

It is noted in Section 5.1 that a TCP-friendly congestion-control algorithm is desirable for a JSNM video system. In this section we first briefly review TCP congestion control and define TCP-friendliness, then we derive JSNM congestion-control design guidelines based on algorithm classification. We present our proposed JSNM congestion-control algorithm and discuss its behavior in Section 6.2.

6.1.1 TCP congestion control

TCP provides congestion control by maintaining a congestion window that controls the number of outstanding unacknowledged packets in the network. On startup, TCP performs *slow start*, during which it doubles the rate for each *round-trip time* (RTT) to quickly gain its fair share of network bandwidth; in steady state, TCP uses an *additive increase, multiplicative decrease* (AIMD) mechanism to probe the network and react to congestion. When there is no indication of congestion, TCP increases window by one slot per RTT; in case of packet loss indicated by a timeout, the window is reduced to one slot and TCP re-enters the slow-start phase. Packet loss indicated by three duplicate ACKs instructs TCP to reduce the window to half its original size.

TCP's throughput T can be approximated by the two following models [65], depending on the network environment, respectively:

1. **Simplified model:** This model does not take into account TCP timeouts:

$$T(t_{RTT}, s, p) = \frac{c \cdot s}{t_{RTT} \cdot \sqrt{p}} \tag{6.1}$$

   where t_{RTT} is the round-trip time, s is the segment size, p is the packet loss rate, and c is a constant value commonly approximated to be $1.5\sqrt{2/3}$.

2. **Complex model:** Equation (6.2) models TCP more accurately in an environment with a high loss rate:

$$T(t_{RTT}, t_{RTO}, s, p) = \min \left\{ \frac{W_m \cdot s}{t_{RTT}}, \frac{s}{t_{RTT}\sqrt{\frac{2bp}{3}} + t_{RTO} \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1 + 32p^2)} \right\} \tag{6.2}$$

where b is the number of packets acknowledged by each ACK, W_m is the maximum size of the congestion window, and t_{RTO} is the TCP timeout interval. This model takes into account rate deductions due to TCP timeouts.

6.1.2 TCP friendliness

TCP friendliness has multiple definitions and is a subject for ongoing debate. In [63], non-TCP flows are defined as TCP friendly when they satisfy the following definition:

Definition 6.1 *Non-TCP flows are TCP friendly when their long-term throughput does not exceed the throughput of a conformant TCP connection under the same conditions.*

A more restrictive definition (for unicast only) can be found in [65].

Definition 6.2 *A unicast flow is considered TCP friendly when it does not reduce the long-term throughput of any coexistent TCP flow more than another TCP flow on the same path would under the same network conditions.*

In this thesis, Definition 6.1 is used for simplicity.

6.1.3 JSNM congestion control design guidelines

Congestion-control schemes can be classified based on various characteristics: window-based versus rate-based, unicast versus multicast, single-rate versus multi-rate, end-to-end versus router-supported, etc. Since JSNM currently addresses only unicast flows, it is confined to single-rate. In addition, based on simple algorithm classification, we conclude that JSNM congestion control should be a rate-based, end-to-end algorithm. We explain our reasoning below.

1. **Window-based versus rate-based:**

   Algorithms belonging to the window-based category use a congestion window with size adjusted based on acknowledgment information from the receiver; algorithms belonging to the rate-based category avoid congestion by dynamically adapting the transmission rate according to some network feedback mechanism that indicates congestion. Our JSNM protocol does not have a packet acknowledgment (at least not in the instantaneous sense) mechanism; thus, it is limited to using a rate-based congestion-control algorithm.

   Rate-based algorithms can further be divided into simple AIMD schemes and model-based schemes. Simple AIMD schemes mimic the short-term behavior of TCP congestion control and display the typical sawtooth like rate fluctuation pattern, which makes AIMD schemes unsuitable for continuous video streaming. Model-based congestion control adapts the sending rate to the average long-term throughput of TCP and produces much smoother rate changes that are better suited to real-time video traffic. They do not mimic TCP's short-term behavior but are still TCP friendly in longer time scales.

   Based on the above discussion, we offer our first guideline about JSNM congestion control:

   Guideline 6.1 *JSNM congestion control should not attempt to emulate the short-term behavior of TCP congestion control, which is detrimental to real-time video transmission. It should adjust the sending rate based on network models that conform to TCP's long-term characteristics.*

2. **End-to-end versus router-supported:**

   Depending on whether or not additional network functionality is required, congestion-control algorithms can be divided into *end-to-end* algorithms and *router-supported* algorithms.

   End-to-end congestion-control algorithms are designed for best-effort IP networks and can be deployed rapidly. They can be further divided into *sender-based* and *receiver-based* approaches depending on which side carries out active congestion control. End-to-end algorithms have the disadvantage of relying on the collaboration of end systems; i.e, they are defenseless against greedy user/applications.

   Router-supported congestion-control algorithms provide additional network capabilities such as feedback aggregation, hierarchical RTT measurements, modification of queueing

strategies. Ultimate fair resource sharing in the presence of unresponsive or non-TCP-friendly flows can only be achieved with router support. However, this mechanism is difficult to deploy because of the high cost in terms of money, time and effort that would have to be spent on Internet infrastructure modifications.

Our JSNM protocol currently does not involve a JSNM-aware router design (although it is certainly an attractive research possibility); it is a transport layer protocol on top of UDP and does not target at modifying existing Internet infrastructure. Thus, we desrive our second guideline about JSNM congestion control:

Guideline 6.2 *A JSNM congestion control algorithm should be an end-to-end, sender-based mechanism that can be readily deployed and does not require additional network functionality support.*

Based on the design guidelines derived above, we designed a JSNM protocol using a simple model-aided, sender-based, single-rate, end-to-end, congestion-control scheme.

6.2 JSNMP Congestion Control

To perform effective congestion control, we need to cope with the bandwidth fluctuation in the network and estimate the currently available network bandwidth based on receiver feedback and a network model. In JSNMP, we use the TCP model developed in [73] and base our network bandwidth estimation technique on the multimedia streaming TCP-friendly protocol (MSTFP) developed by Zhang et al. [74], which attempts to *minimize the number of future packets that are likely to be dropped* and smooth the sending rate based on the observed *packet-loss ratio* P_{loss}, round-trip time, and timeout (TO).

6.2.1 JSNMP packet types

Our JSNMP employs three major types of packets: message packets, video data packets, and feedback packets. The message packets carry instructions and messages between sender and receiver and facilitate session-related tasks such as initialization and termination. The video data packets are the delivery vehicle for actual compressed video; the header of every video data packet includes the *sequence number* (SEQ), intra-inter packet RS coding parameters, the *timestamp* (TS) indicating the time at which the packet is sent. The feedback packets are sent

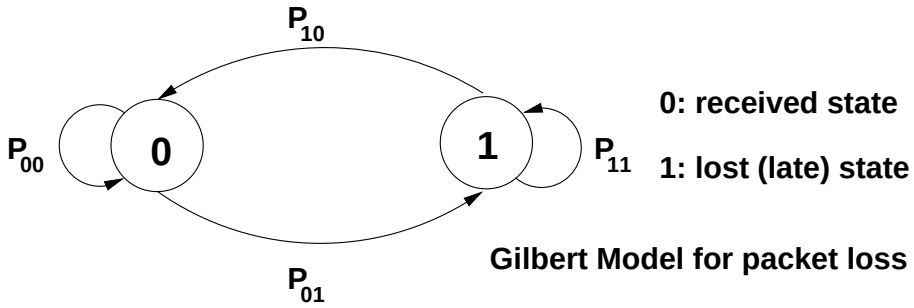

Figure 6.1 Two-state Markov (Gilbert) model for packet loss.

by the receiver at regular intervals, they include a 2-bit flag for each video data packet from the sender (for details refer to Appendix E.3) and information necessary for estimating RTT and TO. When the receiver generates the feedback packet, it acknowledges the latest received video data packet by feeding back its sequence number (SEQ_L), timestamp (TS_L), and the time duration this packet spent in the receiver (ΔRT). The 2-bit flag can reflect one of the four possible states for a packet at the receiver: *received and corrected*, *received but corrupt*, *received too late or lost*, and *other cases*.

6.2.2 Network packet loss model

In a network video-transmission system that includes possible wireless links, a packet could reach its destination but be corrupted by wireless fading, or be lost in the network due to network congestion or timeouts. We simplify the problem by assuming that packet loss due to congestion and packet corruption due to transmission errors are independent. We use a packet-loss model to determine the currently available bandwidth and sending rate, and use the packet-loss rate and channel symbol-error rate to determine the appropriate interpacket and intrapacket RS protection configurations.

We use the two-state Markov model (Gilbert model) to model packet loss in the network. This model is able to capture the dependence between consecutive states. For packet losses, we assume that the transmitted packets are represented by an indicator array $\{S_i\}_{i=1}^{M}$, where $S_i = 1$ indicates that the ith packet has arrived successfully and 0 otherwise. The current state S_i of this stochastic process (shown in Figure 6.1) depends only on its previous state S_{i-1}. We define the transition probabilities between these two states as follows:

$$P_{01} = \text{Prob}\ [\ S_i = 1 \mid S_{i-1} = 0\ ] \tag{6.3}$$

$$P_{10} = \text{Prob}\ [\ S_i = 0 \mid S_{i-1} = 1\ ] \tag{6.4}$$

The maximum-likelihood (ML) estimators of P_{01} and P_{10} given an indicator array $\{S_i\}$ can be expressed as:

$$\hat{P}_{01} = \frac{m_{01}}{m_0} \tag{6.5}$$

$$\hat{P}_{10} = \frac{m_{10}}{m_1} \tag{6.6}$$

$$P_L = \frac{\hat{P}_{10}}{(\hat{P}_{01} + \hat{P}_{10})} \tag{6.7}$$

where m_0 and m_1 are the number of zeroes and ones in the indicator array, m_{01} is the number of times in the indicator array when 1 follows 0, m_{10} is the number of times in the indicator array when 0 follows 1, $m_0 + m_1 = M$, where M is the length of the indicator array, and P_L is the probability that the current state is the loss state. The value of M determines the amount of memory in the estimation; small M allows for faster response to network changes and large M prevents inaccurate estimation due to small-duration network burst variations. In our simulation we set M to be equal to L, the number of packets sent for one GOP.

6.2.3 Estimation of available bandwidth

We estimate the available bandwidth based on information such as RTT, TO, and P_L. RTT can be estimated by the sender based on the feedback from the receiver, using the TCP's RTT calculation equation:

$$RTT = \alpha_1 \times RTT_O + \alpha_2 \times (TS_{now} - TS_L - \Delta RT) + \alpha_3 \times RTT_{tcp} \tag{6.8}$$

$$1 = \alpha_1 + \alpha_2 + \alpha_3 \tag{6.9}$$

where RTT_O is the previously estimated round-trip time, RTT is the current estimate for the round-trip time, RTT_{tcp} is the round-trip time estimate from the TCP feedback link (α_3 is set to zero if an alternative reliable protocol instead of TCP is used to transmit the feedback), TS_{now} is the timestamp indicating the time at which the feedback packet is received at the sender,

and α_1, α_2, and α_3 are weight parameters that are set to 0.72, 0.25, and 0.03, respectively. We adopt the exact same method as in TCP ($TO = \beta \times RTT$) to estimate timeout TO without actually retransmitting any data.

After obtaining the necessary parameters, the sender can estimate the present available bandwidth/throughput X as follows:

$$X(RTT, TO, N, P_L) = \frac{N}{RTT \times \sqrt{2\ \frac{P_L}{3}} + 3 \times TO \times P_L \times \sqrt{3\ \frac{P_L}{8}} \times \sqrt{1 + 32\ P_L^2}} \tag{6.10}$$

where N is the size of the video data packet in bytes. Note the similarity between Equation (6.10) and the TCP throughput expression in Equation (6.2). Our throughput model is actually a simplification of the original TCP model, we define X as the *TCP-estimated throughput*.

6.2.4 JSNM congestion-control algorithm

Because of the delay constraint for video communication, JSNM does not provide an acknowledgment and retransmission scheme. Lost packets cannot be resent. In addition, real-time video applications suffer greatly from large quality variations. Therefore, we must be careful when modifying the packet transmission rate. For example, the current TCP congestion-control algorithm, which uses the AIMD scheme independent of the lost fraction and adjusting interval, will certainly not suit our purpose. Thus, we choose an alternative method to adjust the sending rate based on [74]. The detailed congestion-control algorithm is explained below.

If ($X_k > T_k$)
  $\theta = (TS_{now}$ - $TS_{last})$ / RTT
  If ($\theta > 2$)
    $\theta = 2$
  else if ($\theta < 1$)
    $\theta = 1$
  $T_{k+1} = T_k$ + (N / RTT) $\times\ \theta$
else
  $T_{k+1} = \gamma \times X_k + (1$ - $\gamma) \times T_k$

where X_k is the current (kth iteration) TCP-estimated throughput, T_k is the throughput for the current (kth) iteration, T_{k+1} is the throughput for the $(k+1)$th iteration, θ is a multiplicative

factor, TS_{last} is the timestamp indicating the time at which previous rate adjustment happened, and γ is the weight parameter (between zero and one) that can be used to adjust traffic smoothness. Using the above rate-adjusting procedure, JSNMP has a smaller local variation on the transmission rate, and is less sensitive to random loss caused by channel error.

Theorem 6.1 *The congestion-control algorithm employed in the JSNM protocol is TCP friendly in its long-term behavior.*

Theorem 6.2 *The congestion-control algorithm employed in the JSNM protocol exhibits smoother rate variations than those of TCP.*

Proof: Assuming the TCP-estimated throughput during congestion ($X_k < T_k$) forms an array of random variables that have an *independent identical distribution* (i.i.d), X_0, ..., X_k, with mean μ and variance σ^2, we have:

$$T_{k+1} = \gamma\, X_k + (1-\gamma)\, T_k \tag{6.11}$$

We can determine our congestion-control algorithm's long-term behavior by calculating the expectation of throughput, $\mathrm{E}\,(T_{k+1})$, when there is congestion:

$$\begin{aligned}
\mathrm{E}\,(T_{k+1}) &= \mathrm{E}\,(\ \gamma\, X_k + (1-\gamma)\, T_k\ ) \\
&= \gamma\ \Big(\ 1 + (1-\gamma) + \ldots + (1-\gamma)^{k-1}\ \Big)\ \mathrm{E}\,(X_k) + (1-\gamma)^k\, \mathrm{E}\,(T_0) \\
&= \Big(1 - (1-\gamma)^k\Big)\ \mathrm{E}\,(X_k) + (1-\gamma)^k\, \mathrm{E}\,(T_0) \\
&\rightarrow \mathrm{E}\,(X_k), \quad \text{for k sufficiently large, } 0 < \gamma < 1
\end{aligned} \tag{6.12}$$

Equation (6.12) indicates that the expected throughput of JSNM congestion-control algorithm converges to that of the TCP-estimated throughput given k sufficiently large and γ between zero and one, thus demonstrating long-term similarity to TCP. Theroem 6.1 is proved.

To demonstrate the smoother rate variation of our JSNM protocol, we can compare the respective variances of our congestion-control algorithm and that of TCP (denote the correlation

between X_k and T_0 as $\mathrm{cov}\,(X, T_0)$, which is independent of k):

$$\begin{aligned}
\mathrm{var}\,(T_{k+1}) &= \mathrm{var}\,(\gamma\, X_k + (1-\gamma)\, T_k) \\
&= \mathrm{var}\,\left(\gamma\, X_k + \gamma(1-\gamma)\, X_{k-1} + \ldots + \gamma(1-\gamma)^{k-1}\, X_0 + (1-\gamma)^k\, T_0\right) \\
&= \gamma^2\,\left(1 + (1-\gamma)^2 + \ldots + (1-\gamma)^{2k}\right)\,\sigma^2 + (1-\gamma)^{2k}\ \mathrm{var}\,(T_0) + \\
&\quad 2\,\gamma\,(1-\gamma)^k\,\left(1 + (1-\gamma) + \ldots + (1-\gamma)^{k-1}\right)\ \mathrm{cov}\,(X, To) \\
&= \frac{\gamma}{2-\gamma}\,\left(1 - (1-\gamma)^{2k}\right)\,\sigma^2 + (1-\gamma)^{2k}\ \mathrm{var}\,(T_0) + \\
&\quad 2(1-\gamma)^k\ \mathrm{cov}\,(X, T_0) \qquad (6.13) \\
&\rightarrow \frac{\gamma}{2-\gamma}\,\sigma^2, \qquad \text{for k sufficiently large, } 0 < \gamma < 1 \qquad (6.14)
\end{aligned}$$

For k sufficiently large and γ between zero and one, the second and third items in the right-hand side of Equation (6.13) go to zero. In addition, the coefficient for the first item is always smaller than one, thus demonstrating that $\mathrm{var}\,(T_{k+1}) < \sigma^2 = \mathrm{var}\,(X_k)$ for k sufficiently large. Theorem 6.2 is proved.

6.3 JSNMP Packet Loss-Error Control

After the congestion-control step determines the current network link capacity and sending rate, the JSNM system needs to perform channel FEC encoding. This section explains in detail how the JSNM protocol controls the effect of packet loss and packet error by jointly matching the source and network based on the current packet congestion-loss probability P_{loss} and wireless symbol-error probability P_{symbol}. We first describe a method to estimate these two probabilities, then establish the optimization problem, and finally propose our local-exhaustive-search algorithm to obtain the optimal set of interpacket and intrapacket RS encoding parameters.

6.3.1 P_{loss} and P_{symbol} estimation

The network packet congestion-loss probability P_{loss} can be easily estimated at the *sender* side by counting the total number of "lost" and "other" flags in the feedback packet indicator array (we consider those packets that have arrived late also lost). The wireless symbol-error probability P_{symbol}, which we use to characterize the wireless links assuming slow fading, can be estimated at the *receiver* side: for each type-1 packet, the receiver keeps count of the number of symbol-errors reported from RS decoding. P_{symbol} is then obtained by averaging the number

of symbol-errors over all symbols, and subsequently transmitted back to the sender via the feedback packet.

6.3.2 JSNM optimization

Instead of adjusting interpacket RS coding parameters (n_i, k_i, t_i) and intrapacket RS coding parameters (N_i, K_i, T_i) using *ad hoc* or adaptive strategies, JSNMP attempts to achieve the optimal end-to-end performance by solving an optimization problem based on the R-D curve of the progressive video source encoder and channel characteristics. First, we impose necessary constraints on these parameters (all parameters listed below are integers unless specified otherwise):

$$T = \sum_{i=1}^{L-1} [\ n_i \cdot (N_i + N_{hdr})\ ] + n_L \cdot (N_L + N_{hdr}) \tag{6.15}$$

$$T_S = \sum_{i=1}^{L-1} [\ k_i \cdot (K_i - 1)\ ] + k_L \cdot (K_L - 1) \tag{6.16}$$

$$n_i = 2^w - 1,\ \ 1 \le i \le L-1,\ 1 \le n_i \le 31 \tag{6.17}$$

$$N_i = 2^W - 1,\ \ 1 \le i \le L-1,\ W = 8,\ N_i = 255 \tag{6.18}$$

$$k_i = 2 \cdot y + 1,\ \ 1 \le i \le L-1,\ 1 \le k_i \le n_i \tag{6.19}$$

$$K_i = 2 \cdot Y + 1,\ \ 1 \le i \le L-1,\ 1 \le K_i \le N_i - 4 \tag{6.20}$$

where T is the total video sending rate determined by the network congestion-control algorithm in Section 6.2.4, N_{hdr} is the length of the header section of a JSNMP packet, T_S is the portion of the sending rate corresponding to video source information (note that there is one checksum byte that has to be deducted), and w, W, y, and Y are all integers. Equation (6.17) and (6.18) indicate that the size of both the packet (in symbols) and layer (in packets) has to be the length of a valid RS codeword (we fix N_i to be 255 and use the notation N in the following sections); Equation (6.20) specifies that a minimum amount of four *safe-guard* protection symbols are always present; this is to avoid complete packet loss due to a single symbol error caused by fast fading. The Lth layer is neither intra-RS encoded or inter-RS encoded, so $K_L = N$, $n_L = k_L$, and the latter two can be all positive integers not limited to valid RS codeword length.

The *expected* end-to-end distortion, E(D), is equivalent to

$$
\begin{aligned}
\mathrm{E}(D) &= \sum_{i=1}^{L-1} P_{layer}(i)\ D_{layer}(i) + \sum_{j=1}^{n_L} P_{packet}(j)\ D_{packet}(j) && (6.21)\\
D_{layer}(i) &= d\left(\sum_{m=1}^{i-1} k_i \cdot (K_i - 1)\right)\\
D_{packet}(j) &= d\left(\sum_{m=1}^{L-1} k_i \cdot (K_i - 1) + j \cdot N\right)\\
P_{layer}(i) &= \prod_{m=1}^{i-1} p_{layer}(m)\ [\ 1 - p_{layer}(i)\ ], \quad i = 1, \ldots, L-1\\
p_{layer}(i) &= \sum_{m=0}^{2t_i} \binom{n_i}{m} p_{packet}(i)^m\ [\ 1 - p_{packet}(i)\ ]^{n_i - m}, \quad i = 1, \ldots, L-1\\
p_{packet}(i) &= p_{loss} + p_{error}(i), \quad i = 1, \ldots, L && (6.22)\\
p_{error}(i) &= \sum_{m=0}^{T_i} \binom{N}{m} {p_{symbol}}^m\ [\ 1 - p_{symbol}\ ]^{N-m}, \quad i = 1, \ldots, L && (6.23)\\
P_{packet}(j) &= p_{packet}(L)\ [\ 1 - p_{packet}(L)\ ]^{j-1} \prod_{m=1}^{L-1} p_{layer}(m)
\end{aligned}
$$

where

1. L is the preset number of packet layers. $D_{layer}(i)$ is the resulting distortion if layer i is the *first* layer that cannot be recovered. $D_{packet}(j)$ is the resulting distortion if the jth packet in layer L is the *first* packet that cannot be recovered. $d(\cdot)$ is the rate-distortion function associated with the video coder that we use.

2. $P_{layer}(i)$ is the probability of the ith layer being the *first* layer which cannot be recovered, $p_{layer}(i)$ is the probability that the ith layer cannot be recovered. The cause of failure to recover a layer can be too many packets lost due to network congestion and transmission errors. For the ith layer, at most $2t_i$ packets can be lost.

3. $P_{packet}(j)$ is the probability of the jth packet (in layer L) being the *first* packet that cannot be recovered; $p_{packet}(i)$ is the probability that any single ith layer packet cannot be recovered, this probability characterizes the channel packet loss rate. It is the sum of the probability of the packet being late or lost in the network p_{loss}, and the probability of an ith layer packet being corrupted by fading errors $p_{error}(i)$. The term p_{loss} can be estimated from the packet-status indicator array in the feedback packet, $p_{error}(i)$ can be

calculated from the symbol-error probability (we assume 8-bit symbols in this thesis, so a symbol is effectively a byte), p_{symbol}, which has to be estimated by the receiver and transmitted back to the sender.

Most of the variables are intermediate values created only for the convenience of presentation. The most important variables are p_{loss}, which characterizes the network congestion status, and p_{symbol}, which characterizes the wireless channel condition. In the event that no wireless links exist, we assume that packets are always received perfectly and the only packet losses are due to congestion; i.e., $p_{packet}(i) = p_{loss}$. Our objective here is to find the optimal set of (N, K_i, T_i) and (n_i, k_i, t_i) that minimizes E(D), within the constraints. Assuming L layers and wireless links, we have $3L - 2$ variables (K_i, n_i, and k_i) to optimize; assuming wireline links only, we have $2L - 1$ variables.

This constrained optimization problem can be solved locally using numerical gradient procedures. However, in this case, given the limited range of possible values for all the RS parameters and the computing power of today's computers, we propose a *local-exhaustive-search* method (denoted as the LES algorithm). The algorithm first initializes all parameters to be the same as those used for the previous GOP, then performs an exhaustive search within a local window of those parameters and finds the optimal set. For first-time system initialization, a fixed set of conservative values are used.

By running simulations using a global-exhaustive-search algorithm and analyzing the results, we derive a suitable set of local window sizes for the LES algorithm: for the intra-RS encoding rate K_i, a window size of ± 16 symbols is used; for the inter-RS encoding parameters n_i and k_i, a global exhaustive search is performed since there are only four possible choices for n_i. Simulations show that this algorithm works well and gives excellent results (see Section 5.4).

If we compare the optimization setup for both the JSCM peer video-transmission system (see Appendix B.1) and the minimax disappointment video broadcasting system (see Appendix C), a fundamental similarity can be observed in the following aspects:

1. All three systems achieve the best *expected* end-to-end performance by using prioritized source bit-stream coupled with unequal error protection techniques.

2. All three systems involve solving a constrained, non-linear, possibly integer-based optimization problem, which can be solved using a gradient-based algorithm (the LES algorithm for JSNM is just a simpler alternative).

3. In all three cases, both the source and channel coders are not modified in any way. Their individual R-D characteristics (source) or error-probability performance curves (channel) are all that is required, thus demonstrating the generality of the matching approach.

CHAPTER 7

CONCLUSIONS

To conclude this thesis, we first briefly recall our original objectives. Based on the acknowledged fact that jointly matching the source and channel encoders can bring performance gains in some communication systems, this thesis applies JSCM to video transmission with different types of channels to achieve the following specific goals: (1) to demonstrate that JSCM truly benefits video transmission under a resource-limited situation; (2) to develop a matching scheme for peer video transmission; (3) to show that JSCM can be efficiently applied to video broadcasting; (4) to extend JSCM to JSNM; and (5) to address the fundamental question of whether the communication engineer should focus his efforts on separate source and channel coder optimization, or on better cooperation (matching) between the source and channel coders.

Our results demonstrate that we have, indeed, achieved all these objectives:

1. JSCM does truly achieve end-to-end performance gains for all three types of channels: single sender-receiver pair, multireceiver broadcasting, and multisender-multireceiver network video transmission. The application of JSCM to video communication is beneficial.

2. Our general JSCM system for peer video transmission functions well regardless of the specific source and channel coder. Additionally, the adaptive system reduces system computational complexity and facilitates the practical application of the JSCM scheme.

3. Video broadcasting also stands to benefit from JSCM by applying the MD performance criterion, which, we believe, has advantages over conventional criteria in that it is universally fair, intuitive, and theoretically meaningful.

4. Successful application of JSCM over networks has been achieved by designing a JSNM transport layer protocol, JSNMP, complete with its specific packet format, congestion-control algorithm, packet loss-error control mechanisms, and session support. Simulations demonstrate superior performance to conventional approaches.

We conclude from our findings that significant gains can be achieved by the simple application of JSCM to existing communication systems, without any modifications to the current video source coders and channel coders. Given the fact that current research progress in separate coder optimization has more or less reached the point of saturation (the performance increase in video source coders are now measured in one tenth of a decibel in PSNR, and channel coders' performance are close to the Shannon limit), significant system-level (end-to-end) performance improvements are most easily achieved by jointly matching the source and channel coders.

Furthermore, while joint source-channel *matching* can be regarded as merely a special case of the more general joint source-channel *coding*, it has certain advantages. The more general JSCC approaches often involve low-level source and channel coder *co-design*, which brings extensive modifications to the original source or channel coders or both. In addition, the resulting JSCC systems usually function only for the specific source-channel coder. On the other hand, JSCM techniques require neither detailed technical knowledge nor extensive modifications to the source and channel coders, and the resulted JSCM systems perform well for a wide range of source-channel coder pairs.

Many interesting problems related with JSCM have not been fully addressed in this thesis because of time limitations; we would like to mention a few interesting possibilities:

1. In our simulation we have made the assumption of slow-fading wireless channels. It would be desirable to find out how well our system performs under fast-fading channels and, if necessary, develop a JSCM system for fast-fading channels as well.

2. ARQ-based techniques can improve performance under noisy channels by ensuring the delivery of video information. However, for channels with high delays, ARQ techniques are at a disadvantage. From our simulation we observe that progressive video stream coupled with unequal error protection using FEC can give excellent results without the help of ARQ, although ARQ-based system does give a slightly better performance under bad channel conditions. Thus, a hybrid ARQ-FEC scheme could potentially perform better. This is certainly a worthwhile research direction.

3. Our current JSNM protocol is a unicast protocol without network support. It is of high interest to determine how to design JSNM system for a multicast scenario and how to design JSNM-aware routers.

It is our hope that this thesis will prove beneficial to other JSCM researchers and stimulate more research interest toward the topic of joint source-channel video transmission.

APPENDIX A

ESTIMATION OF R-D CHARACTERISTICS

The rate-distortion characteristic function for a particular video encoder is usually difficult to obtain because of the following three major issues:

1. The complexity of the coder and lack of an accurate model, which makes the derivation of an analytical expression for the R-D function or even its approximated form highly difficult.

2. The high volume of raw data involved, which eliminates the possibility of using exhaustive *real-time* estimation techniques because of the computational load required.

3. The R-D function's video-content-dependency, which prevents the R-D function (obtained off-line) for one sequence to be used on the sequence being processed.

Various approaches have been proposed to address these issues.

1. Instead of trying to derive the exact or approximate expressions of the R-D functions, researchers try to estimate the functions using exhaustive procedures and fit them to a universal parametric model such as sum of exponentials or order-n polynomials [21]. This approach provides sufficient accuracy despite the lack of a theoretic model.

2. To reduce the computational complexity, instead of exhaustively estimating the R-D function in its entire range, several "control points" can be defined and their R-D values estimated [75]. The points in between can be either obtained by interpolation/extrapolation or directly calculated from the parametric model.

3. To solve the content-dependency problem, the following procedure can be used [76] [77]:

    a. A large set of typical sample sequences are classified into a number of representative categories, or "eigen-categories."

b. For each eigen-category, the R-D functions for each sequence in the category are estimated; the average of these functions is then defined as the R-D function for this eigen-category. The R-D functions for all the eigen-categories are then assumed to form a basis for all common sequences.

c. For any new sequence out of the sample set, it is projected onto the basis using sequence-identification based or similar methods; its R-D function can be represented using the basis R-D functions.

One of our design objectives for the JSCM system is *generality*, namely, that it can be applied to a wide variety of source-channel coder pairs. To achieve this end, the JSCM system should require as little detailed low-level technical knowledge of the encoder as possible, and should estimate the rate-distortion characteristics of the encoder in real time. Our proposed approach is to model the video source encoders as a "black box" with rate-adjusting switches; we perform source encoding-decoding at several predetermined "control points" on the rate-distortion curve, and obtain the entire curve via cubic-spline interpolation between these points.

Here we discuss, in detail, how to perform the real-time rate-distortion estimation, using several commonly used video coders as examples. Although attempts have been made to address candidates from each of the major video coder categories, this is certainly by no means a complete list. We demonstrate the following examples in the hope that R-D estimation for other coders would be essentially similar.

A.1 Motion-JPEG Encoder

Motion-JPEG encodes every single video frame in JPEG format and uses a frame-specific *quality factor q* to adjust the coding rate. Figures A.1, A.2, and A.3 show the rate-distortion curves for the "Football," "Miss America," and "Tennis" video sequences containing 20 frames. Experimental evidence revealed that the curves are very similar for all the frames within the same GOP, given there is no scene change; this fact greatly simplifies the estimation of the rate- distortion characteristics for the Motion-JPEG coder. Now we only have to perform rate-distortion estimation once and then use the obtained curve for the rest of the current sequence until we detect a scene change.

The computational complexity involved in obtaining the distortion for the several "control points" can be further reduced by recognizing the following facts:

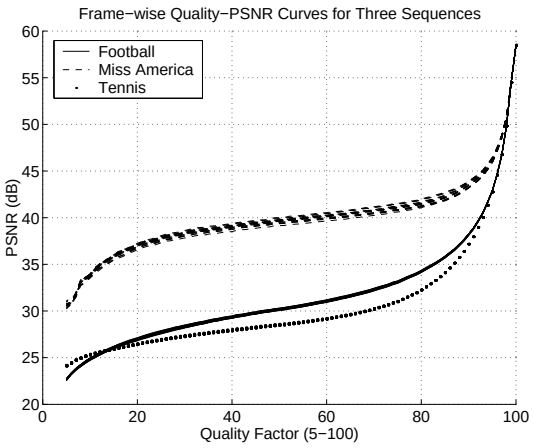

Figure A.1 PSNR versus quality-factor.

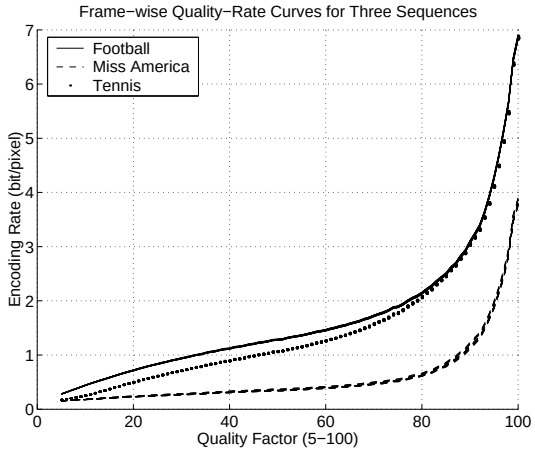

Figure A.2 Rate versus quality-factor.

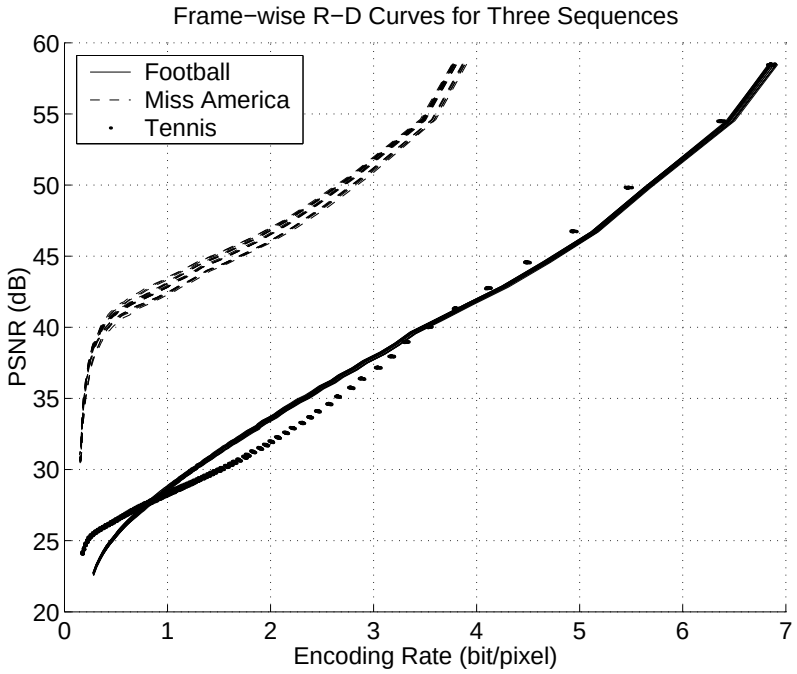

Figure A.3 Motion-JPEG rate-distortion curves.

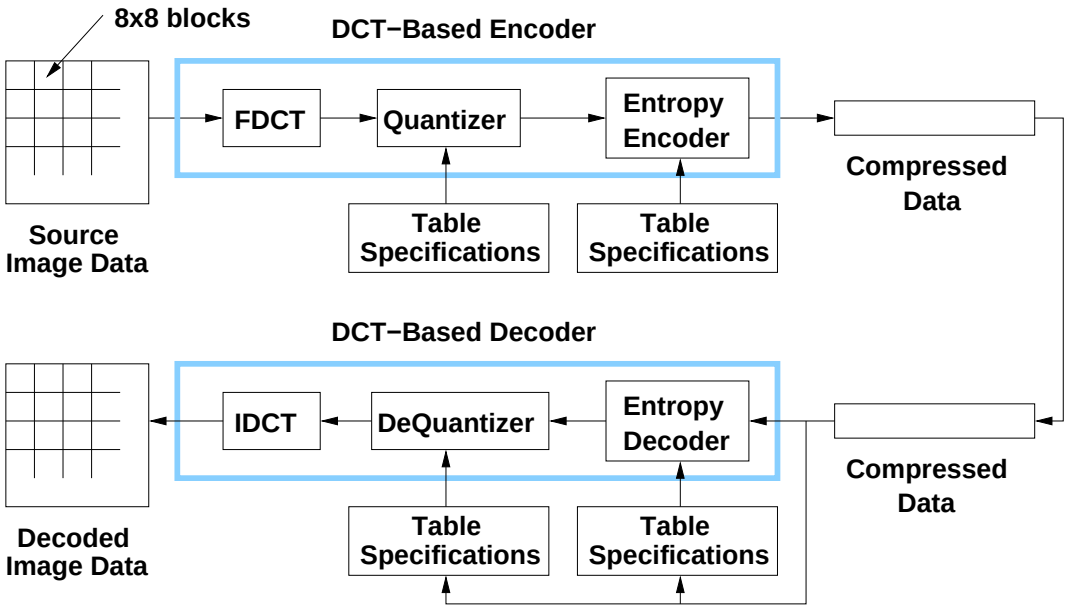

Figure A.4 The JPEG codec block diagram.

1. The most computationally expensive operation involved in JPEG encoding (shown in Figure A.4) is the 2-D DCT operation, which needs to be performed *only once* for each sequence in our R-D estimation.

2. There is no need to perform the equally expensive inverse 2-D DCT transform because distortion can be calculated from the DCT coefficients directly, thanks to the unitary property of the DCT.

3. For a size $N \times M$ frame, if we want to obtain K "control points," the required operations are K 2-D quantization of size $N \times M$, plus K entropy coding operations.

After the initial estimation is performed, the rest of the curve can be obtained by piecewise cubic interpolation between the "control points." It is observed in simulation that at most five "control points" are required to achieve a high degree of approximation accuracy. Furthermore, the JPEG encoder seldom operates at the $q > 85$ and $q < 15$ region for compression efficiency and image quality concerns; thus, the R-D curves within the region where we are mostly interested in are almost linear. With the help of chip-level Motion-JPEG instruction support, real-time implementation of this estimation procedure is feasible.

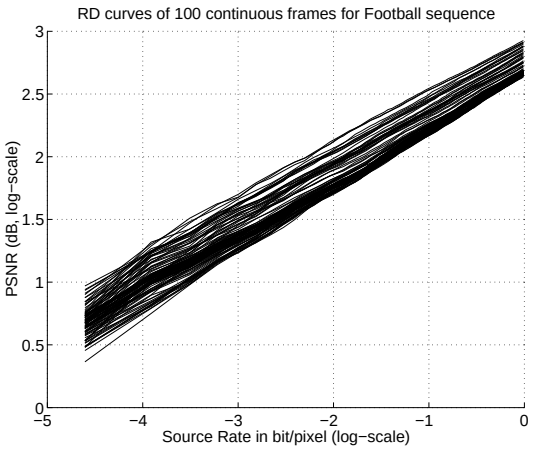

Figure A.5 Framewise R-D curves.

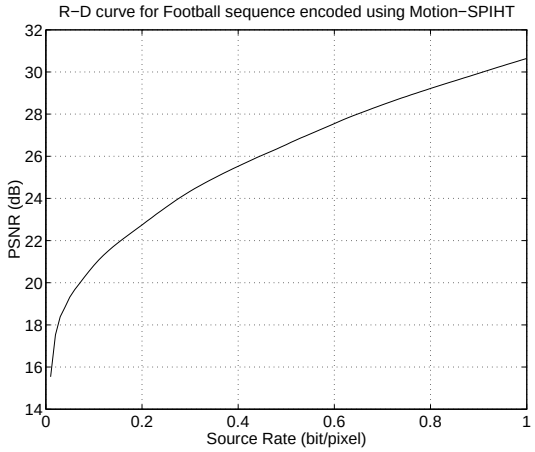

Figure A.6 Frame-averaged R-D curve.

A.2 Motion-SPIHT Encoder

The collection of framewise R-D curves for frames in a Motion-SPIHT-encoded video sequence are no longer close to each other. However, it is observed (from Figure A.5) that, when plotted in a log-log scale, R-D curves for frames in a Motion-SPIHT-coded video sequence can be approximated by a collection of converging linear functions. The R-D function for any particular frame is thus determined by a single point on the curve, which is readily available when we perform the initial source compression. In fact, because SPIHT is progressive and its underlying wavelet transform is unitary, distortions can be calculated accumulatively in the encoding process and a single run of the encoder could yield multiple points from the R-D curve, thus increasing approximation accuracy. The corresponding computation requirement is essentially neglegible.

The frame-averaged R-D curve for the Motion-SPIHT encoder is shown in Figure A.6.

A.3 Conditional Block Replenishment (CBR) Encoder

As mentioned in Chapter 2.2, the CBR coder has many different ways to adjust its encoding rate. To simplify encoding, we choose N_b, the number of blocks to be coded and sent, as the encoding rate parameter. Figure A.7 shows the typical relation between N_b and the source rate. Similar to the Motion-JPEG coder, its R-D curve estimation can also be simplified by removing frame-dependency (this is not true if we use T_b, the "modified" block MSE threshold as the

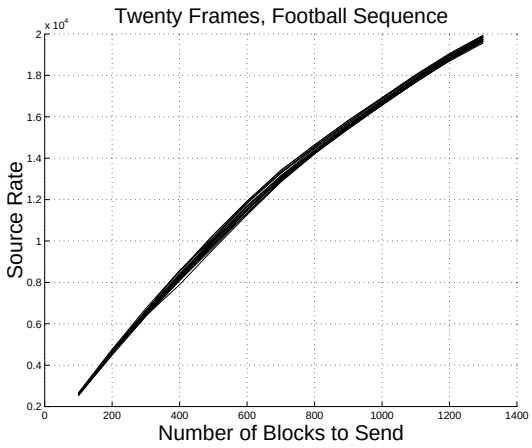

Figure A.7 Blocks versus source rate.

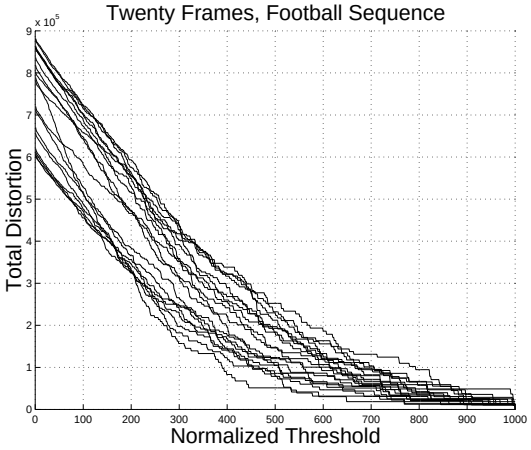

Figure A.8 Distortion versus threshold.

encoding rate parameter, as can be observed from Figure A.8, where curves are obviously not close to each other.)

A.4 3D-SPIHT Embedded Encoder

The bitstream produced by a 3D-SPIHT encoder is almost-strictly progressive and has a typical rate-distortion curve as shown in Figure A.9. The three different R-D curves in the plot are obtained in the following fashion: the "original" curve is the true R-D curve generated by evaluating the 3D-SPIHT encoder at different rates; the "truncated" R-D curve is obtained by decoding a fixed-rate encoded bit-stream truncated at locations corresponding to different rates; the "nontruncated" R-D curve is obtained similarly to the "truncated" version except that instead of truncating at location i, it creates a bit error at location i and keeps the entire bit-stream for decoding. The closeness of the "truncated" and "nontruncated" R-D curves indicates that 3D-SPIHT produces a highly prioritized bit-stream. If we use either one of these two curves as an approximation of the "original" 3D-SPIHT R-D curve, then similarly to Motion-SPIHT, the R-D function for a 3D-SPIHT coder can be obtained in a single run of the coder at a fixed rate.

A.5 H.26x Coder

H.26x generally uses a *macroblock quantization* (MQUANT) factor to adjust coding rate. Figures A.10 and A.11 show the approximate relationship between the average MQUANT factor

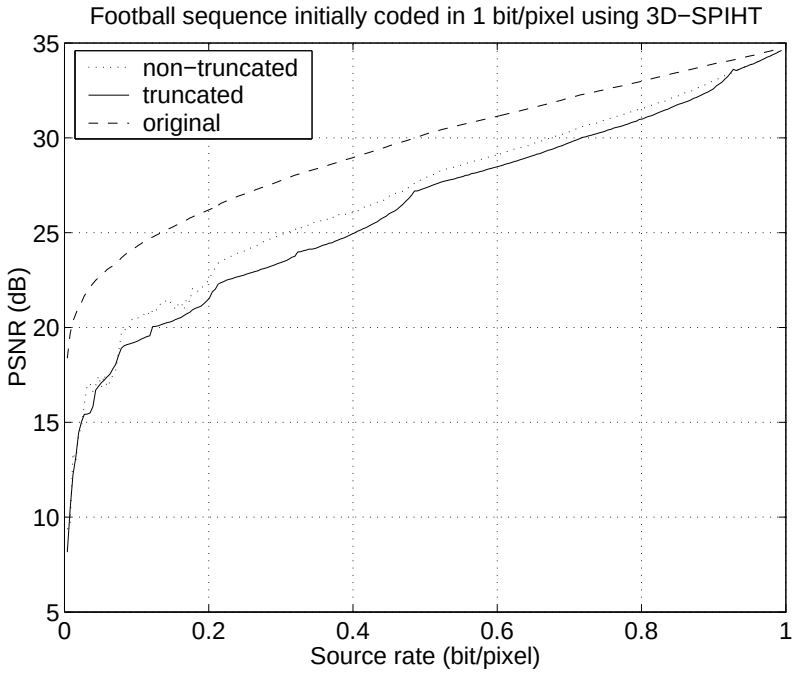

Figure A.9 3D-SPIHT RD curve.

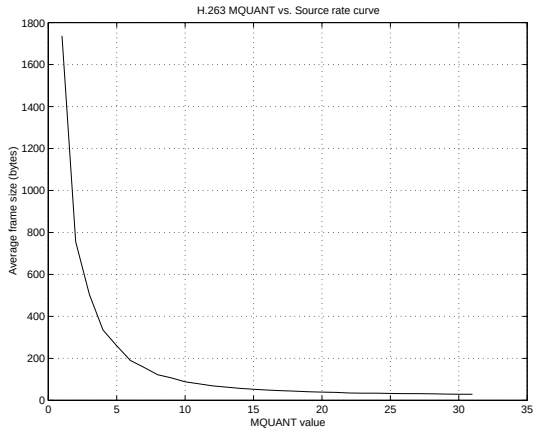

Figure A.10 Source rate versus MQUANT.

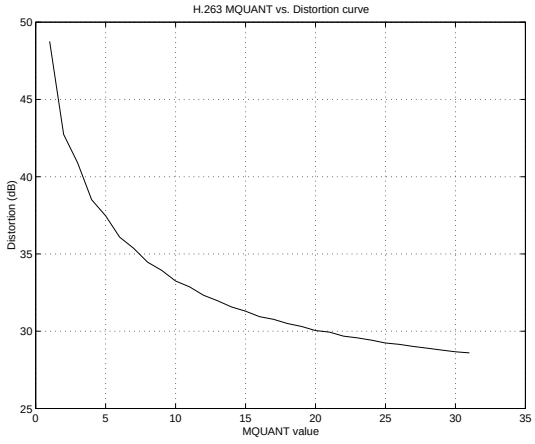

Figure A.11 Distortion versus MQUANT.

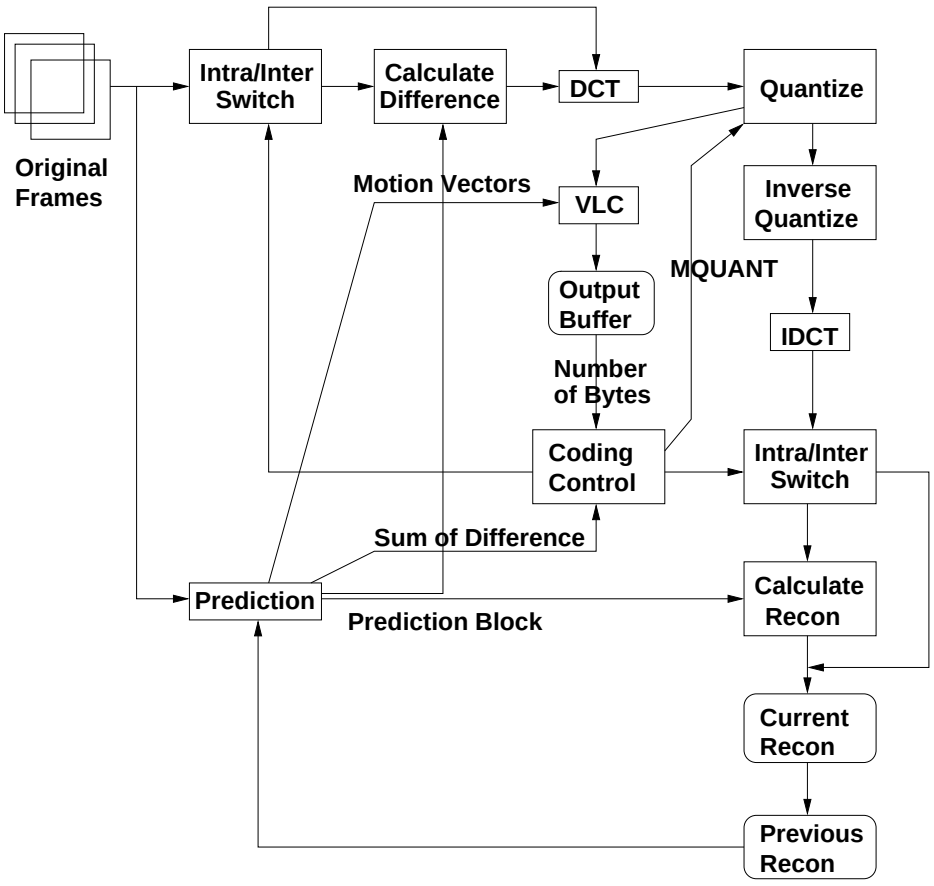

Figure A.12 H.263 encoder diagram.

and the source rate or the decoding distortion. However, because of the much higher complexity of the H.263 encoder and the online rate-control algorithm it applies, it is no longer practical to estimate its rate-distortion characteristics on-line. Instead, we developed an adaptive JSCM system for this type of coder; for details please refer to Appendix B. A block diagram for the H.263 encoder is shown in Figure A.12.

A.6 Layered H.263 Source Coder

The H.26x series of video coders do not produce progressive bit-streams. In order to create a pseudoprogressive property for coders like these, we propose a layered H.263 video encoder. This layered H.263 encoder encodes the same video sequences at different rates and then simply concatenates the compressed sequences together to form the layered stream. Each layer is an independent H.263-encoded sequence targeted for a specific performance level, typically in a descending order. Upon the reception of the signal, users decode the layer which gives the least

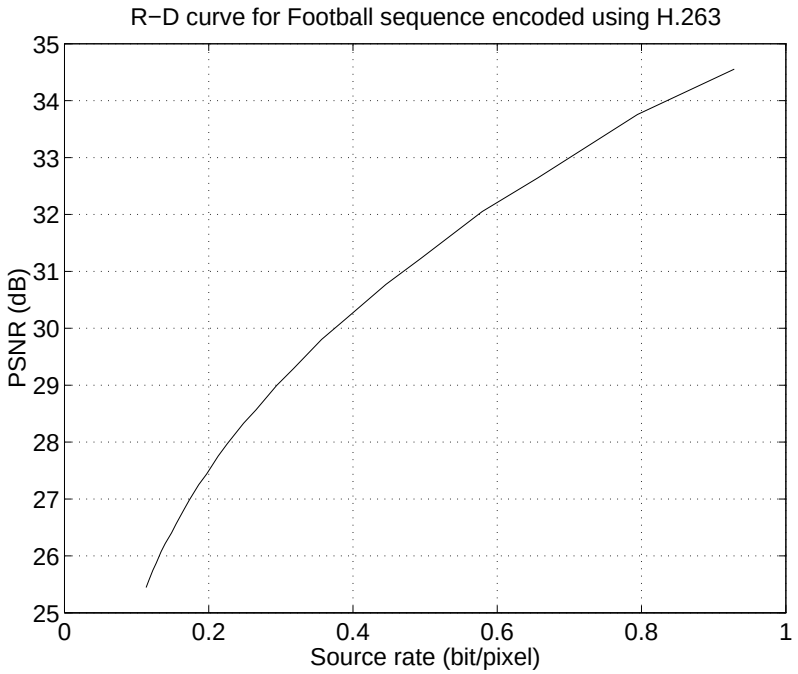

Figure A.13 Rate-distortion curve for the H.263 encoder.

decoding distortion. Figure A.13 shows the output distortion of an H.263 encoder working at different rates. The layered H.263 encoder simply locates the corresponding rate to achieve a certain distortion level and uses these rates to create the layers.

We note that this layered H.263 encoder is a highly inefficient multiresolution coder, with layers that are totally independent and cannot enhance each other. Furthermore, simulation results show that the inefficiency of retransmitting coarse information at fine layers renders any layer-configuration with more than three layers wasteful and impractical. However, the system is simple and exploits the increased compression efficiency of the advanced H.263 coder.

APPENDIX B

JSCM PEER TRANSMISSION SYSTEMS

B.1 General System

The general JSCM system developed in our early work has been tested on various source-channel coder combinations. Here we describe the Motion-JPEG and RS coder combination as a detailed example; the other cases are essentially similar.

To formulate the problem when the source coder is Motion-JPEG, we assume that we have a series of video frames that are divided into GOPs of size N.

We define the optimization problem as minimizing the final end-to-end distortion with respect to two size-N vectors Q and P, where $Q = (q_1, q_2, ..., q_N)^T$ contains the quality-factors used to JPEG-encode each frame, and $P = (p_1, p_2, ..., p_n)^T$ contains the amount of protection we use for transmitting all the packets contained in a certain frame. Because the source rate and distortion are both functions of the quality factor q, we define the source model to be two sets of curves: the distortion versus quality-factor curve $d_i(q_i)$ and the source-rate versus quality-factor curve $s_i(q_i)$ (i is the frame index).

However, as we mentioned in Chapter 2, further investigation into the problem revealed that the $d_i(q_i)$ and $s_i(q_i)$ curves are very similar for all the frames in the same GOP (see Figure A.3). We can use the same quality factor and amount of protection for all the frames in the same GOP, provided that the buffer size limitation is also satisfied. Though this yields a slightly suboptimal solution, this approach is comparable in performance with the truly optimal solution but is much simpler in terms of computational complexity because we reduced the $2N$-D optimization problem to a 2-D problem.

The final optimization problem can be formulated as finding the optimal scalars q and p that minimize the final distortion D:

$$D = \frac{1}{N}\sum_{i=1}^{N} D(i) \tag{B.1}$$

$$D(i) = P_{fsucc}(p) \times d(q) + P_{ffail}(p) \times d_{conceal}(i) \tag{B.2}$$

$$P_{fsucc} = \prod_{n=1}^{M(q)} P_{psucc} \tag{B.3}$$

$$M(q) = \frac{s(q)}{L} \tag{B.4}$$

Here D is the final end-to-end distortion, $D(i)$ is the expected distortion for the ith frame, $P_{fsucc}(p)$ and $P_{ffail}(p)$ are the frame transmission success and failure probabilities when we use p protection symbols for each block, $d(q)$ is the distortion for each frame when coded using quality-factor q, $d_{conceal}(i)$ is the distortion when we use the $(i-1)$th frame in place of the ith frame, $M(q)$ is the number of packets in each frame, $s(q)$ is the frame length when coded using quality-factor q, and L is the predefined RS packet size.

We must also satisfy the following decoding buffer-size constraints:

$$b_i = max(b_{i-1} + s(q) + p - R, 0) \tag{B.5}$$

$$b_i \leq b_{max}, i = 1, 2, ..., N \tag{B.6}$$

The optimization problem we have defined is constrained and nonlinear; furthermore, the variables can only take integer values. In order to solve this problem, we need to transform it into an unconstrained problem. We chose the penalty function approach [78], [79] and modified our cost function as:

$$D = \sum_{i=1}^{N} D(i) + C \times \sum_{i=1}^{N} \max(0, b_i - b_{max}) \tag{B.7}$$

where $D(i)$ is defined in Equation (B.2) and the extra term is the penalty function; when C goes to infinity, this forces the solution to converge to the optimal solution to the constrained problem. (Because the solution to our problem is discrete, the converged solution is the truly optimal answer, not a suboptimal one as in the continuous case.) In practice, we use a sufficiently large value for C.

Figure B.1 shows the detailed system implementation. The system complexity is of the same order as DCT coding, and usually no more than three iterations are required to find the

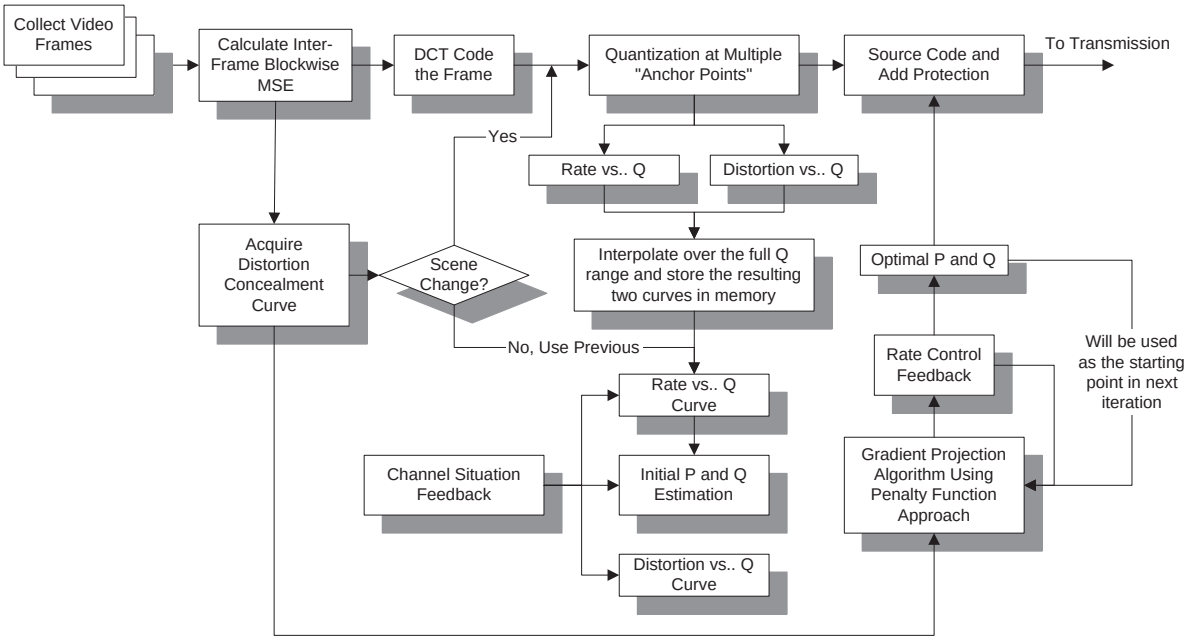

Figure B.1 Motion-JPEG and RS: JSCM system implementation.

optimal configuration. For further details on issues such as system initialization, please refer to the author's Master's thesis [44]. For the reader's reference, the system implementation diagram for the CBR-RS coder combination is attached here in Figure B.2.

B.2 Adaptive System

For the adaptive system introduced in Section 3.3, we choose H.263 as the source encoder and RCPC codes as the channel coder. The simplified system diagram is shown in Figure B.3:

As shown in the figure, the raw video frames are first input to the H.263 video encoder, the resulting intra and inter frames are then packetized and CRC-protected before they are sent to the RCPC channel encoder. The joint matching system adjusts the source and channel rate by modifying the MQUANT factor and RCPC puncture rate for *each packet* using the algorithm described in Section 3.3, the optimal puncture rate is estimated from periodic channel feedback, and the MQUANT value is predicted from the current buffer occupancy using H.263's internal rate-control algorithms.

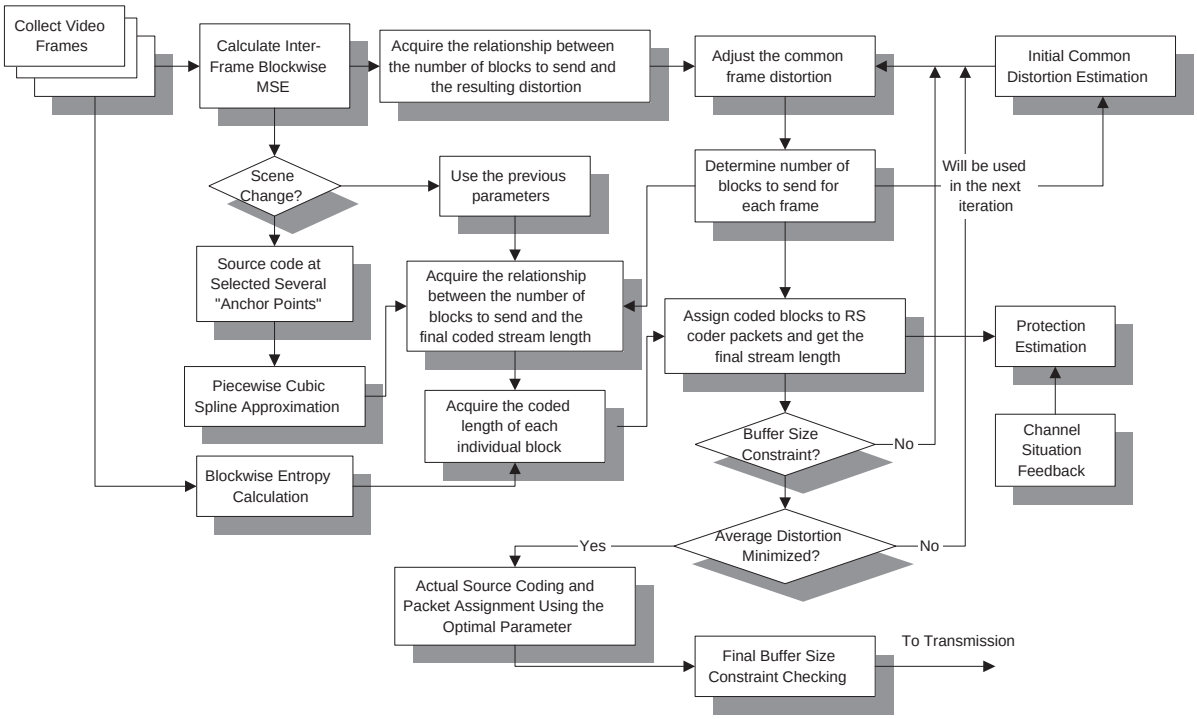

Figure B.2 CBR and RS: JSCM system implementation.

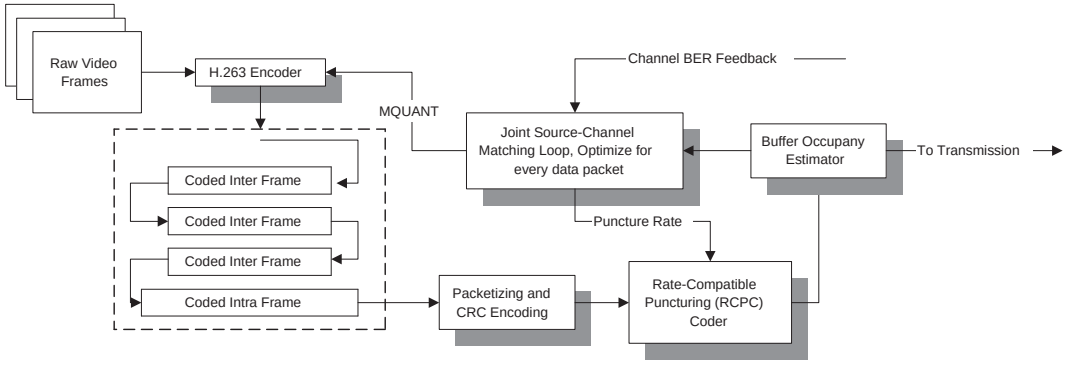

Figure B.3 Simplified system for H.263 and RCPC encoder.

APPENDIX C

JSCM BROADCASTING SYSTEMS

In this section we explain in detail our four simulation systems and the underlying mathematical optimization problems.

C.1 Motion-SPIHT and RCPC: Rate-Constrained

First we consider the case where the channel capacity is limited. We choose Motion-SPIHT as the source encoder and RCPC as the channel encoder.

The minimax optimization problem can be shown mathematically as follows:

$$K_i = \sum_{j}^{J} m_{ij} \tag{C.1}$$

$$s_i = \sum_{j}^{J} m_{ij}\ r_j\ p \tag{C.2}$$

$$D_{i,l} = \sum_{k=1}^{K_i} \left[P_{k+1,l} \sum_{n=1}^{k} (1 - P_{n,l}) \right] d_i(k) \tag{C.3}$$

$$D_{0,l} = \min_{m_{ij}} \left(\frac{\sum_i^N D_{i,l}}{N} \right) \tag{C.4}$$

$$P = \max_{l,m_{ij}} \left(D_{0,l} - \frac{\sum_i^N D_{i,l}}{N} \right) \tag{C.5}$$

$$b_i = \max(b_{i-1} + s_i - R, 0) \tag{C.6}$$

$$b_i \leq b_{max},\ i = 1, 2, \ldots, N \tag{C.7}$$

where K_i, m_{ij}, and r_j are defined as before; p is the fixed packet size in bytes; s_i is the total length (source plus protection) for the ith frame; $D_{i,l}$ is the expected received performance for the ith frame by the lth user; and $d_i(k)$ is the quality of the decoded ith frame if the kth packet

is the first packet lost. $P_{n,l}$ is the transmission failure probability of the nth packet for the lth user (the m_{ij} packets in the jth packet group would have $P_{n,l}$ equal to the error probability for user l when using RCPC protection rate r_j), $D_{0,l}$ is the expected optimal performance for user l, which can be obtained by solving Equation (C.4), and N is the size of the video sequence. P is the maximum of the disappointment, the value we need to minimize. Note that we solve (C.4) and (C.5) as two separate optimization problems; the solution of (C.4) gives us $D_{0,l}$, while the solution of (C.5) gives us the optimal m_{ij} and minimax disappointment P.

A particular problem related with video transmission, called rate control [80], must also be addressed. Since video transmission usually allows only a small, finite delay, and the decoder usually has a finite decoding buffer size and constant decoding flow, we should avoid buffer overflow or underflow. If we use b_i to denote the buffer occupancy at the ith time index and R to denote the constant decode flow, we should ensure that b_i is never greater than b_{max}, the maximum decoding buffer-size. Our optimization becomes a constrained problem because of this particular requirement and the total channel capacity constraint.

One way to incorporate the rate-control constraint is to modify the cost function P by adding a penalty item associated with buffer overflow:

$$P' = P + C \times \sum_{i=1}^{N} \max(0, b_i - b_{max})$$

where C is a large constant. When C goes to infinity, the nonconstrained solution converges to the constrained solution; in a discrete optimization setting, they will be equal given C sufficiently large.

To solve this optimization problem, we initialize the values for m_{ij} with the expectation that they are close to the optimal values, and employ a gradient-based procedure to derive the optimal solution. Usually at least six to seven iterations are required for the algorithm to converge, but since broadcasting parameter optimization is usually done off-line, we can afford the cost.

C.2 3D-SPIHT and RCPC: Rate Constrained

In this case we substitute the Motion-SPIHT coder in the previous simulation system with the 3D-SPIHT video encoder. The same encoding procedure still applies except that now instead of encoding on a frame basis, we encode and transmit the entire GOP in one step. We

do not have framewise rate-control issues with this transmission scheme; on the other hand, a total delay of one GOP is introduced.

Similar to the previous case, the minimax optimization problem can be shown mathematically as follows:

$$K = \sum_{j}^{J} m_j \tag{C.8}$$

$$s = \sum_{j}^{J} m_j \; r_j \; p \tag{C.9}$$

$$D_l = \sum_{k=1}^{K} \left[P_{k+1,l} \sum_{n=1}^{k} (1 - P_{n,l}) \right] d(k) \tag{C.10}$$

$$D_{0,l} = \min_{m_j} D_l \tag{C.11}$$

$$P = \max_{l,m_j} (D_{0,l} - D_l) \tag{C.12}$$

where the entire 1 bit/pixel encoded 3D-SPIHT bitstream is first divided into packets of size p, and only the first K are sent; The K packets are further divided into J groups, each containing m_j packets; each packet group is assigned a particular RCPC encoding rate r_j; $P_{n,l}$ is again the transmission failure probability of the nth packet for the lth user; $d(k)$ is the decoding distortion if the kth packet is the first packet lost; D_l is the expected distortion for the lth user; $D_{0,l}$ is the expected minimal distortion for the lth user; and P is our minimax disappointment. Equation (C.11) defines the single-user JSCC optimization problem, and Equation (C.12) defines the minimax disappointment optimization problem for this broadcasting system. They can be solved using exactly the same technique used in the previous case.

C.3 Layered-H.263 and RCPC: Rate Constrained

In this case we again consider the situation when we have a limited channel capacity. We choose layered-H.263 as the source encoder. The encoding procedure can be described as follows:

1. Obtain all layers by performing multiple compression using predetermined layer-specific compression parameters.

2. Apply RCPC protection with different rates to each layer.

3. Concatenate the encoded sequences together and transmit.

This minimax problem can be mathematically expressed as follows:

$$\sum_{i}^{N} s(q_i) r_i \leq C \tag{C.13}$$

$$D_k = \sum_{i}^{N} \prod_{j}^{i-1} P_k(q_j, r_j) (1 - P_k(q_i, r_i)) d(q_i) \tag{C.14}$$

$$\log(P_k(q_i, r_i)) = A_k s(q_i) r_i + B_k \tag{C.15}$$

$$D_{0,k} = \min_{Q,R} D_k \tag{C.16}$$

$$P = \max_{k,Q,R} (D_{0,k} - D_k) \tag{C.17}$$

where N is the number of layers, q_i is the ith quality factor we use to compress the sequence, $s(q_i)$ is the resulting compressed sequence length, and r_j is the RCPC coding rate for the ith layer. The sum of the length of all layers with channel protection should be less than the total channel capacity. $P_k(q_j, r_j)$ is the *layer-error probability* for the jth layer and kth user; we approximate its value using a log-affine function given in (C.15) (the channel SNR for the kth user is then hidden inside parameter A_k and B_k, which need to be preestimated before optimization) [12]. The term $d(q)$ is the decoding distortion when using quality factor q. Q and R are the quality factor vector $(q_1, q_2, \ldots, q_N)$ and the RCPC coding rate vector $(r_1, r_2, \ldots, r_N)$. Equations (C.16) and (C.17) again define the joint source-channel optimization problem for a single user and the minimax optimization problem for the entire broadcasting system. They can similarly be solved using gradient-descent-based algorithms.

C.4 3D-SPIHT: Power Constrained

Unlike the previous three cases, this time we consider the case where the total system transmission power is limited. We employ 3D-SPIHT as the source coder and adjust the transmission power for each bit in order to achieve unequal protection. The transmission bit-power profile is shown in Figure C.1 and the transmission procedure can be described as follows:

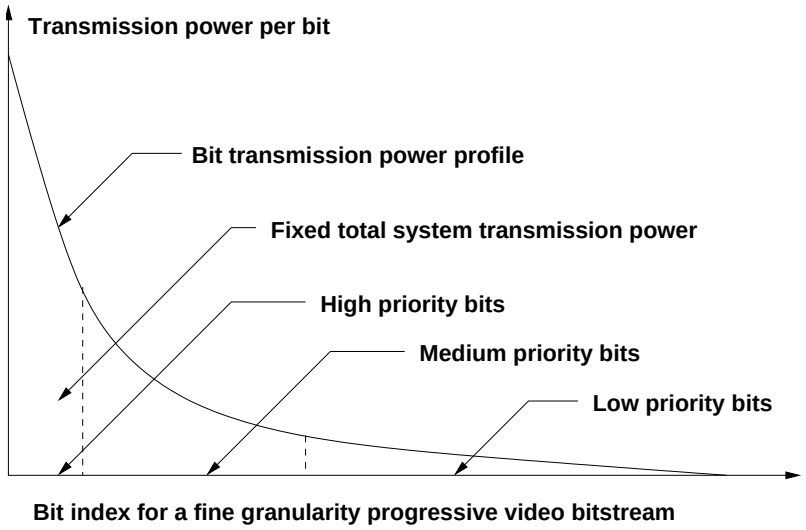

Figure C.1 Power-constrained system.

1 Raw video frames are first divided into groups of size 16 and compressed using the 3D-SPIHT video encoder; its rate-distortion curve is then estimated.

2 The compressed video source stream is transmitted with energy per bit optimized under the total energy constraint.

The minimax optimization problem can be cast mathematically as follows:

$$\sum_{j}^{L_s} e_j \leq E_{total} \tag{C.18}$$

$$p_k(e_j) = Q\left(\sqrt{\frac{2e_j}{N_{0,k}}}\right) \tag{C.19}$$

$$D_k = \prod_{j=1}^{L_s} (1 - p_k(e_j))\, D_c + \sum_{i=1}^{L_s} \left[\prod_{j=1}^{i-1} (1 - p_k(e_j))\right] p_k(e_i)\, d(i) \tag{C.20}$$

$$D_{0,k} = \min_{E} D_k \tag{C.21}$$

$$P = \max_{k,E} (D_{0,k} - D_k) \tag{C.22}$$

where E_{total} is the total limit on system transmission power, $p_k(e_j)$ is the transmission error probability for the jth bit for the kth user, given bit energy e_j and channel noise $N_{0,k}$; D_k is

the resulting distortion for the kth user; L_s is the total number of bits and D_c is the distortion caused by source compression. The term $d(i)$ is essentially the rate-distortion curve for the current video stream; it is the decoding distortion if the ith bit is the first bit in error (we discard all bits after it); and E is the bit transmission power profile vector $(e_1, e_2, \ldots, e_{L_s})$. $D_{0,k}$ is the expected optimal performance for the kth user, which can be obtained by solving (C.21), and we obtain P, the maximum disappointment value for all users, by solving (C.22).

We try to minimize P by finding the optimal energy allocation vector E $(e_1, e_2, \ldots, e_{L_s})$. This minimax optimization problem can be similarly solved by the first method of successive approximation. Rate-control is not a concern in this case since we always encode a group of frames as an entity and do not start decoding until we receive the entire stream. This entails an initial delay of one group of frames, which is acceptable in typical video-conferencing applications.

APPENDIX D

MINIMAX OPTIMIZATION THEORY

In this chapter we elaborate on the performance and convergence issues of our algorithms used to solve the minimum disappointment minimax [81] optimization problem. We focus on the case of a power-constrained system.

The standard minimax optimization problem with constraints can be stated as follows: Let $f_i(X), i \in [0:N], X = (x_1, \ldots, x_n)$ be functions defined and continuously differentiable on some open set $\Omega' \subset E_n$; let Ω be a convex closed (not necessarily bounded) subset of Ω'. The objective is to find a point $X^* \in \Omega$ such that

$$\max_{i\in[0:N]} f_i(X^*) = \inf_{X\in\Omega} \max_{i\in[0:N]} f_i(X)$$

Consider the function defined on Ω'

$$\phi(X) = \max_{i\in[0:N]} f_i(X)$$

The problem we have set is precisely to minimze the function $\phi(X)$ on the set Ω. Also define the following set $R(X)$

$$R(X) = \{i \in [0:N] \mid f_i(X) = \phi(X)\}$$

The necessary (and sufficient for convex cost functions) conditions for a minimax solution is stated in the following theorem [81]:

Theorem D.1 *A necessary condition for a point $X* \in \Omega$ to be a minimum point of $\phi(X)$ on Ω is that*

$$\inf_{Z\in\Omega} \max_{i\in R(X^*)} \left(\frac{\partial f_i(X^*)}{\partial X}, Z - X^* \right) = 0$$

If $\phi(X)$ is convex, this condition is also sufficient, and the point X^* is called a stationary point of $\phi(X)$ on Ω.

Condition D.1 is equivalent to the following theorem [81]:

Theorem D.2 *In order that $\phi(X)$ have a minimum on E_n at a point X^*, it is necessary, and if $\phi(X)$ is convex, also sufficient that*

$$\inf_{\substack{g\in\bar{\Gamma}(X^*)\\ \|g\|=1}} \max_{i\in R(X^*)} \left(\frac{\partial f_i(X^*)}{\partial X},\ g \right) \geq 0$$

Theorems D.1 and D.2 essentially state that we can apply gradient-descent type algorithms to solve the minimax optimization problem. Based on the above theorems, we employ the first method of successive approximations (a direct generalization of the steepest-descent algorithm) to solve the minimax optimization problem. For details of the algorithm please refer to [81]. We discuss the convexity of our $\phi(X)$ function in the following section.

D.1 Convexity of the Cost Function

The necessary and sufficient conditions for a minimax solution stated in the previous section requires that the cost function $\phi(X)$ be convex, which is equivalent to the convexity of $f_i(X)$, based on Theorem D.3. Thus, it is of great interest to investigate the convexity of $f_i(X)$. In showing $f_i(X)$ is convex, we also prove that the joint source-channel optimization problem for a single user class has a convex cost function, and thus various gradient-descent algorithms can be used with a guarantee of convergence to a global minimum.

We make the following observations:

1. The rate-distortion functions for both the Motion-SPIHT and the 3D-SPIHT encoders are *continuous, monotonic, and convex* (in the MSE sense). This is not strictly true in a real situation, but it is a reasonable assumption to make.

2. The error probability versus bit-energy function of an AWGN channel is expressed by the Q function.

We revisit the mathematical expression for our optimization problem and relate it with the minimax optimization theorems:

$$D_k = \prod_{j=1}^{L_s} (1 - p_k(e_j))\, D_c + \sum_{i=1}^{L_s} \left[\prod_{j=1}^{i-1} (1 - p_k(e_j)) \right] p_k(e_i)\, d(i) \tag{D.1}$$

$$f_k(E) = D_{0,k} - D_k, \quad E = (e_1, e_2, \ldots, e_{L_s}) \tag{D.2}$$

$$\phi(E) = J = \max_{k,E} (D_{0,k} - D_k) \tag{D.3}$$

Based on Assumptions 1 and 2, both the Q function and $d(i)$ are continuous, twice differentiable functions. We show that D_k is a convex function by first calculating the gradient of the distortion D_k with regard to the lth energy factor e_l:

$$\begin{aligned} \frac{\partial D_k}{\partial e_l} &= -D_c \frac{\partial p_k(e_l)}{e_l} \prod_{j=1, j\neq l}^{L_s} [1 - p_k(e_j)] + d(l) \frac{\partial p_k(e_l)}{\partial e_l} \prod_{j=1}^{l-1} [1 - p_k(e_j)] \\ &\quad - \sum_{i>l}^{L_s} \left\{ \frac{\partial p_k(e_l)}{\partial e_l} \prod_{j=1, j\neq l}^{i-1} [1 - p_k(e_j)] \right\} p_k(e_i) d(i) \qquad \text{(D.4)} \\ &= \frac{\partial p_k(e_l)}{\partial e_l} \Bigg\{ - \prod_{j=1, j\neq l}^{L_s} [1 - p_k(e_j)]\, D_c + \prod_{j=1}^{l-1} (1 - p_k(e_j)) d(l) \\ &\quad - \sum_{i>l}^{L_s} \left[\prod_{j=1, j\neq l}^{i-1} (1 - p_k(e_j)) \right] p_k(e_i) d(i) \Bigg\} \qquad \text{(D.5)} \end{aligned}$$

Denote the item in the braces as A (note that A does not depend on e_l) and take the second-order derivative and we get:

$$\frac{\partial^2 D_k}{\partial e_l^2} = \frac{\partial^2 p_k(e_l)}{\partial^2 e_l} A \tag{D.6}$$

so the convexity of D_k is determined by convexity of $p_k(e_l)$. We next observe the following:

$$p_k(e_j) = Q\left(\sqrt{\frac{2e_j}{N_{0,k}}} \right) \tag{D.7}$$

$$\frac{\partial p_k(e_j)}{\partial e_j} = \frac{1}{\sqrt{2\pi} N_0}\, e^{-\frac{e_j}{N_0}}\, {e_j}^{-\frac{1}{2}} \tag{D.8}$$

$$\frac{\partial^2 p_k(e_j)}{\partial {e_j}^2} = \frac{1}{\sqrt{2\pi} N_0}\, e^{-\frac{e_j}{N_0}}\, {e_j}^{-\frac{1}{2}} \left(-\frac{1}{N_0} - \frac{1}{2e_j} \right) \tag{D.9}$$

We note that the second derivative of $p_k(e_j)$ is always negative when e_j is positive, thus $p_k(e_j)$ is convex over this region, and so is D_k. Following Theorem D.3 [81], $\phi(E) = J$ is thus convex.

Theorem D.3 *Let Ω be a convex set. If all $f_i(X)$ are convex on Ω, then $\phi(X)$ is also convex on Ω.*

D.2 Simplification of the Optimization Problem

In the power-constrained case, since we are optimizing with regard to each bit, the size of the optimization problem can become relatively large when we have a large number of bits to transmit. To reduce the computational complexity, we can make the following two simplifications.

The first simplification comes from an observation during simulation. We observe that when we have a reasonable number of user classes (more than seven), with channel SNRs relatively spread out, the optimal energy allocation vector for the minimax optimization problem is approximately a linear combination of all the optimal energy allocation vectors for the joint source-channel coding problem for each individual user class. Thus, we can reduce the dimension of the problem down to less than 10 variables by solving for the linear scaling factors. This significantly reduces computational complexity.

For the second simplification, instead of adjusting the transmission power for each bit, we group bits into blocks and adjust the transmission power on a block basis. Using blocks of bits means a lesser computational load but possibly larger minimax disappointment since we have lost part of the flexibility to adjust. A reasonable block size is essential to the system performance (a typical value could be 8, for a byte). Furthermore, we need to show that the cost function is still convex. Assuming a block size of N, we make the following modifications to the original expressions:

$$L_B = \left\lfloor \frac{L_s}{N} \right\rfloor \tag{D.10}$$

$$P_k(e_j) = \sum_{i=1}^{N} \binom{N}{i} p_k(e_j)^i \left(1 - p_k(e_j)\right)^{(N-i)}$$

$$D_k = \prod_{j=1}^{L_B} \left(1 - P_k(e_j)\right) D_c + \sum_{i=1}^{L_B} \left[\prod_{j=1}^{i-1} \left(1 - P_k(e_j)\right) \right] P_k(e_i)\, d(iN) \tag{D.11}$$

where N is the bit-block size, L_B is the total number of blocks, $P_k(e_j)$ is the *block* transmission error probability if each bit inside this block is transmitted with power e_j (we assume that the total block is lost if even a single bit is in error and ignore the location of the error).

Similarly, we can show that the convexity of the cost function depends on the convexity of $P_k(e_j)$. We can calculate the second derivative of $P_k(e_j)$ as follows:

$$\frac{\partial P_k(e_j)}{\partial e_j} = N\frac{\partial p_k(e_j)}{\partial e_j}(1-p_k(e_j))^{N-1} \tag{D.12}$$

$$\frac{\partial^2 P_k(e_j)}{\partial {e_j}^2} = N(1-p_k(e_j))^{N-2}\left[\frac{\partial^2 p_k(e_j)}{\partial {e_j}^2} - (N-1)(\frac{\partial p_k(e_j)}{\partial e_j})^2\right] \tag{D.13}$$

$$\frac{\partial^2 p_k(e_j)}{\partial {e_j}^2} < 0 \tag{D.14}$$

thus we conclude that the second derivative of $P_k(e_j)$ is always negative, so the cost function remains convex.

APPENDIX E

A JSNM PROTOCOL

The JSNM protocol is a transport-layer protocol that tries to achieve best end-to-end network video transmission quality by jointly optimizing the video source encoder and network link. It contains a congestion-control mechanism to regulate video traffic and an error-control algorithm to combat packet loss due to congestion and transmission errors. The sender performs the task of network estimation based on feedback information, JSNM optimization, video stream segmentation, intrapacketizing and interpacketizing, and transmission; the receiver handles video stream reassembly, RS decoding, and creating feedback packet. The feedback packet is transmitted back to the sender via a separate TCP link.

This is a very primitive protocol used for demonstration purposes only. Many secondary implementation details are not addressed for the time being. Significant refinements are required in order for this protocol to be used in a real-life application.

E.1 Format of JSNM Data Packets

The following diagram (Figure E.1) shows the basic format of a JSNM packet embedded in a UDP packet. As mentioned in previous chapters, there are three types of JSNM packets:

1. **Protected source information packet:** This type of packet contains a segment of encoded video source information, one checksum byte, plus intrapacket RS protection bytes.

2. **Interpacket RS protection packet:** This type of packet contains only interpacket RS protection bytes.

3. **Unprotected source information packet:** This type of packet contains only encoded video source information and its checksum byte with no intrapacket RS protection, nor does it have any parallel packets containing interpacket RS protection.

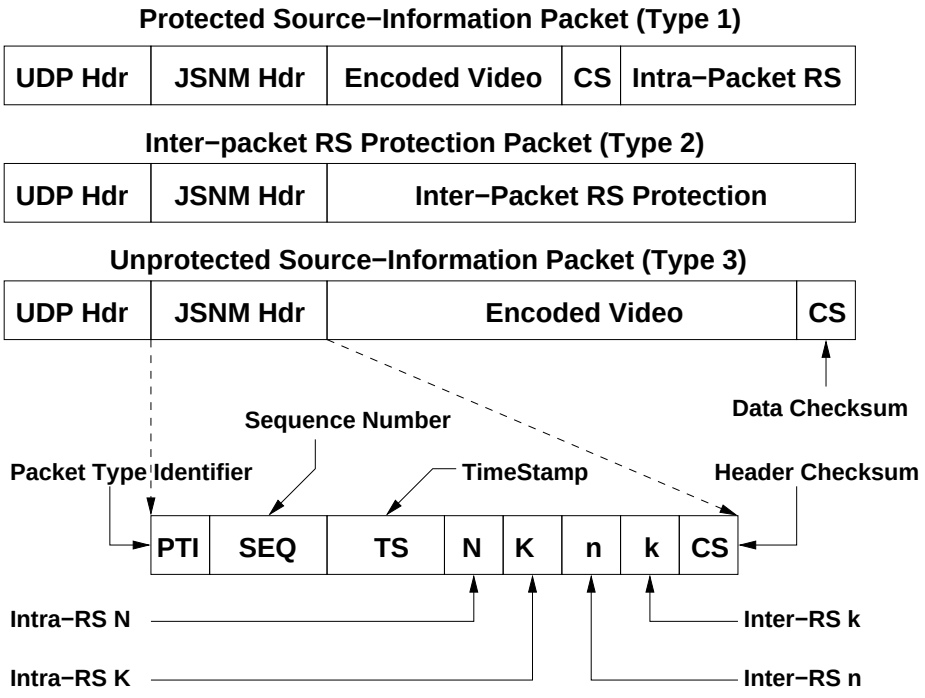

Figure E.1 Format of JSNM packets.

All packets are sent in the form of a UDP packet, so there is a leading UDP header, followed by a JSNM header section, which contains 10 bytes (throughout this thesis we define a byte as an octet):

1. Packet header identifier (PTI): The PTI byte identifies the type of current packet. Its value is 255 for packet types 1 and 3, 0 for packet type 2.

2. Sequence number (SEQ): The SEQ is a double-byte integer that records the current packet's sequence number to facility receiver reassembly.

3. Timestamp (TS): The TS is a double-byte integer that records the current packet's timestamp in milliseconds. The receiver uses this information to estimate the round-trip time.

4. N, K, n, k (RS Parameters): These single-byte values are intra and inter RS protection parameters (hence the reason why the value of N and n are upper-limited by 255.) Note that N is actually redundant since it is fixed to be 255 and known to the receiver. It is provided here for potential future variable-length packet support.

5. Header checksum (CS): The JSNM header contains crucial information for decoding, so it is imperative to know if the header is correctly received. If not, efforts must be made to recover the correct header.

The following rules are employed in generating the header section:

1. The two values for the PTI byte, 255 and 0 (binary expressions: 11111111 and 00000000) are chosen to be as distant as possible from each other in a Hamming sense.

2. The initial sequence number SEQ_0 for a video session are generated randomly. For the following packets: $SEQ_{i+1} = (SEQ_i + 1) \bmod (65535)$.

3. The timestamp TS has two bytes, the first of which represents *seconds*, the second of which represents *milliseconds*. We assume that the sender and receiver synchronize their system clock beforehand in the session initialization process.

4. The values for all the RS encoding parameters are positive integers less than 256. More specifically:

    a. Since the value of n (the interpacket RS codeword length) has to be equal to $2^m - 1$, where m is an integer, a valid value for n has a binary form of a series of consecutive 1s. Also, for all practical uses, the value of n will not exceed 31. (Otherwise, the receiver needs to wait for more than 63 packets before it can start RS decoding, which introduces too much delay).

    b. The intrapacket RS codeword length N is redundant for the time being. It is currently set to be identical to PTI as extra redundancy protection.

    c. For type-2 packets, N and K are not actually meaningful since no intrapacket RS protection is used within those packets. They are set to be the same values as in the type-1 packets associated with the current class to provide even more redundancy protection.

    d. For type-3 packets, none of the RS coding parameters is used. N is set to the same value as PTI; K is set to the total number of type-3 packets, and n and k are concatenated together to keep a copy of SEQ.

5. The CS header checksum byte is generated using one's complement arithmetic.

E.2 Recovery of JSNM Packet Header

Due to possible transmission errors, JSNM packets may arrive at the receiver corrupted. Although JSNM does provide RS protection at both the intrapacket and interpacket level, this protection scheme could not possibly function without knowing the correct original encoding configurations, which is contained in every JSNM packet header. Thus, it is crucial to recover the header information before any further operations can be performed. This section explains in detail how to recover the headers.

E.2.1 Checksum verification

The very first step in header recovery is to verify the header checksum CS byte. A *trust coefficient*, t_i, is associated with each packet header and initialized to be 1. A checksum mismatch indicates that errors have occured and t_i is reduced by half. A matching checksum leaves t_i unmodified.

E.2.2 Packet type identification

The second step is to identify packet types from the two copies of PTI bytes stored in the header. Based on the maximum-likelihood estimation principle, a simplistic approach can be used: if the PTI byte has more 1s than 0s, then it is identified to be a (un)protected source information packet; otherwise it is identified to be an interpacket RS protection packet. The complete decision logic can be described as the following:

If $PTI = N$ (most probably no errors)
    If PTI has more 1s than 0s
        $PTI = 1$
    else
        $PTI = 0$
else $PTI \neq N$ (there are errors)
    If both PTI and N have more 1 than 0
        $PTI = 1$
    else if both PTI and N have more 0 than 1
        $PTI = 0$
    else if $PTI + N$ has more 1s than 0s

$PTI = 1$

else if $PTI + N$ has more 0s than 1s

$PTI = 0$

else

PTI postponed for later decision

We will revisit the packet header identification step later, when we have additional information from the rest of the header bytes. For example, it is impossible for a type-1 packet to be between two type-2 packets. This kind of logic constraint could be used later to correct identification errors.

E.2.3 Sequence number recovery

In a network channel, it is possible for packets to arrive at the destination out of order. To correctly reconstruct the video bit stream, we need to utilize the sequence number information included in the header section. Thus, recovering the sequence number is vital to the reassembly procedure.

The receiver allocates an indicator array, F, whose element indicates whether or not a specific packet has arrived. The receiver also keeps two counters: C_c, which keeps the latest received sequence number; C_e, which keeps the next *expected* sequence number. C_e is initialized to be the value sent by the sender REQ message, C_c is set when the first packet arrives. Under normal situations, $C_e = SEQ$, but this could be invalidated when any packet arrives out of order. More specifically, the following situation could happen:

1. $C_e = SEQ$: the current received packet sequence number matches the receiver's expectation. This is a normal packet arrival.

2. $C_e < SEQ$: the received sequence number is greater than the receiver's expected value. If the corresponding indicator flag in the F array is not set, this most probably means a *future* packet has arrived out of order; if the flag is already set, this means this sequence number has been corrupted.

3. $C_e > SEQ$: the received sequence number is lower than the receiver's expected value. this means the sequence number has been corrupted.

Depending on which situation occurs, the receiver takes corresponding actions based on the following strategy:

1. $C_e = SEQ$: The corresponding flag element in the F array is set. Both C_e and C_c are increased by 1, and the packet content is sent to an internal buffer for further processing.

2. $C_e < SEQ$: The receiver checks the corresponding element in the F flag array.

    a. If the flag is not set, then a *future* packet has arrived. The flag is set, the packet content buffered, and $C_c = \max(C_c, SEQ)$. C_e is not modified.

    b. If the flag is set, then either the current SEQ or the previous SEQ of the same value if bogus. In this case, we employ the first-arrived-stays (FAS) strategy and declare the current SEQ corrupted.

3. $C_e > SEQ$: Based on the FAS strategy, either this SEQ is corrupted or this packet belongs to the previous GOP. This packet is immediately discarded and not buffered.

When the SEQ for the current received packet is declared to be corrupted, it must be recovered to prevent the current packet from being discarded. The receiver keeps all SEQ-corrupted packets in the memory and notes down their arrival time. When the time arrives to play back the current received GOP, the receiver will do its best to make intelligent and reasonable guesses of the SEQ values. Only when this fails will the receiver discard the packet. The receiver recovers the SEQ values under the guidance of the following rules:

1. **Recovery based on neighboring SEQ:** If an SEQ-corrupted packet physically arrived after packet $i-1$, and before packet $i+1$, then the corrupted SEQ is recovered as i. The same rule applies to multiple (up to 3) consecutive SEQ-corrupted packets.

2. **Recovery based on neighboring TS:** The packet sequence number SEQ and its timestamp TS are closely related. Clearly a packet with a smaller sequence number must also have an earlier timestamp. This rule can also be used in conjunction with the previous rule to increase the correctness of recovery.

3. **Recovery based on RS requirement:** Since the interpacket RS codeword length n has to be equal to $2^m - 1$, we need to collect $2^m - 1$ packets before we can start interpacket RS decoding, among which k packets would be type-1 packets ($PTI = 1$) and $n - k$ packets

would be type-2 packets ($PTI = 0$). Thus, if we have $k-1$ type-1 packets from the same layer (which can be derived from the values of K, n, and k in the header section) and $n-k-1$ type-2 packets from the same layer, plus one type-1 packet and one type-2 packet, all from the same layer but arrived out of order and do not follow the recovery-by-neighborhood rule, we can still safely recover their sequence number by filling the blanks in the RS codeword.

4. **Failed Recovery:** When both the two above situations do not apply, we discard the current packet. Discarding packets is *not* a fatal event in the JSNM protocol because of the intrapacket RS decoder's ability to correct packet erasures. We can safely discard $n-k$ packets in a layer and still recover everything. As a matter of fact, *if we already have k packets with confirmed SEQ for a layer, we should discard all of the rest of the packets (for the same layer) with corrupted SEQ numbers to reduce the probability of error.*

E.2.4 Timestamp recovery

Recovering packet timestamp information is important for the receiver to correctly estimate the RTT. Fortunately, it is relatively easy to recover the timestamp information, since the sender transmits video packets at a constant rate; thus, the timestamps should be evenly spaced. Timestamp recovery is performed at the same time as sequence number recovery since they complement each other.

E.2.5 Interpacket RS configuration recovery

After the receiver receives a series of packets, identifies their packet types and recovers their sequence numbers, the next step is to determine the packet layer boundary. All the packets from the same layer contain the same K, n and k, this could be used to identify packets from the same layer. Furthermore, a layer boundary (n) can only occur at packet $2^m - 1$.

After the receiver receives the first packet and verifies its header checksum, the receiver fetches the value of n and, if it is a valid RS codeword length, uses it as the temporary layer boundary. A subsequently arrived packet either confirms the value of n, or submits another valid RS codeword length $\hat{n}$ as a candidate. Each candidate for the value of n has an associated scalar v which is increased whenever this candidate is confirmed by a newly arrived packet (we denote that as a *vote*). The candidate with the highest vote becomes the final n.

The value of k is determined by the same candidate-vote strategy. It is different from n in that it does not have the $2^m - 1$ limitation. However, it must be smaller than the final n and be an odd integer.

E.2.6 Intrapacket RS configuration recovery

The value of K for packets in the same layer can be obtained using the candidate-vote procedure. It must be smaller than 256 and be an odd integer.

E.2.7 Second-iteration header recovery

After the previous steps, the packet headers have been recovered, and the receiver could conceivably go directly to the video stream reassembly procedure. However, an additional second-iteration header recovery has proved to be helpful and to improve recovery accuracy.

1. **Correcting illegal PTI values:** The packet layer structure does not allow certain cases such as a type-2 packet in between two type-1 packets, or vice versa. In other words, packets of the same type should always form a consecutive array with a valid length: a type-1 packet array has odd length while a type-2 packet array has even length, and their sum should be a valid RS codeword length. These rules can be used to find irregularities in the PTI values and correct them.

2. **Removing unnecessary packets with dubious SEQ:** As mentioned before, if the receiver has already received k packets (for the same layer) whose SEQ numbers are confirmed (not based on intelligent guesses), then it is essentially unnecessary to recover the other packets (from the same layer) with corrupted SEQ (or small trust coefficient t_i). We can simply discard them and treat them as RS erasures.

3. **Correcting irregular RS configurations:** In Section 5.3 we pointed out that there are certain relations between the RS configuration parameters for different layers. For example, both the intrapacket RS encoding rate K_i/N and the interpacket RS encoding rate k_i/n_i should normally increase with layer index i, etc. We can correct any irregularities at this stage by going back to the RS configuration recovery step and select the candidate with the second-highest vote.

The recovery of the JSNM packet header is now complete. Based on the header information, the receiver can now reassemble the video bit-stream.

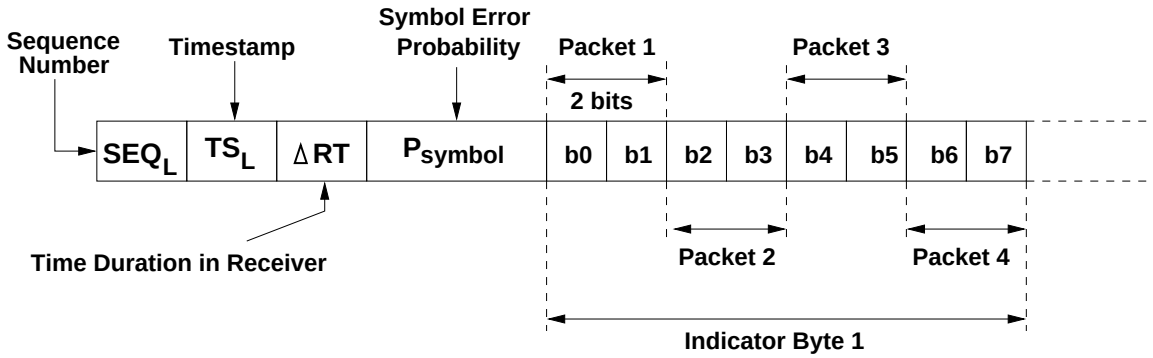

Figure E.2 Format of the JSNM feedback packet.

E.3 Format of the JSNM Feedback Packet

Another important data structure in the JSNM protocol is the format of the feedback packet, which contains the information required for the receiver to estimate the current network link condition. It is shown in Figure E.2.

The JSNM feedback packet contains a 10-byte header and an array of bytes indicating the status of each packet sent by the JSNM sender. The header consists of SEQ_L, the sequence number for the latest received packet, TS_L, its original timestamp, ΔRT, the time duration this packet has spent in the receiver, and P_{symbol}, the estimated wireless symbol error probability. The existence of both SEQ_L and TS_L provides redundancy protection, in case one of them is not correctly received, the sender can still derive the correct timestamp from the other one. In the indicator array, two bits in each byte are allocated for each packet, so the total length of the JSNM feedback packet in bytes is the total number of packets divided by 4. The value of the two bits and its corresponding indicated packet status are:

1. **00:** The "correct" flag, indicating that the packet has arrived in time and can be correctly decoded.

2. **01:** The "corrupted" flag, indicating that the packet has arrived in time but cannot be correctly decoded (too many errors have occurred, SEQ cannot be recovered, etc).

3. **10:** The "lost" flag, indicating that the packet has not arrived within the specified deadline. The receiver will regard any packet that arrives after the deadline as a lost packet.

4. **11:** The "other" flag, mostly reserved for type-3 packets. When the receiver finishes receiving packets, it is impossible for it to tell if the sender has sent out more type-3

packets, which have not arrived in time because they are delayed or simply lost in the channel. The receiver sets the "other" flag for these packets up to the end of the feedback packet. This flag can also be used on type-1 and type-2 packets to indicate that a packet has arrived too late *and* cannot be correctly decoded.

The length of the JSNM feedback packet is predetermined by the sender according to the target rate. The sender transmits this information to the receiver via the REQ message in the session negotiation stage.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[2] R. E. V. Dyck and D. J. Miller, "Transport of wireless video using separate, concatenated, and joint source-channel coding," *Proceedings of the IEEE*, vol. 87, pp. 1734–1750, 1999.

[3] B. Belzer, J. Liao, and J. D. Villasenor, "Adaptive video coding for mobile wireless networks," in *Proceedings of ICIP 1994*, vol. 2, pp. 972–976.

[4] J.-R. Li, S. Ha, and V. Bharghavan, "HPF: A transport protocol for supporting heterogeneous packet flows in the internet," in *Proceedings of IEEE Infocom 1999*, vol. 2, pp. 543–550.

[5] S. B. Z. Azami, P. Duhamel, and O. Rioul, "Joint source-channel coding: Panorama of methods," in *Proceedings of CNES Workshop on Data Compression 1996*, pp. 1232–1254.

[6] G. M. Davis and J. M. Danskin, "Joint source and channel coding for image transmission over lossy packet networks," in *SPIE Conference on Wavelet Applications of Digital Image Processing XIX 1996*, vol. 2847, pp. 376–387.

[7] K. Ramchandran, A. Ortega, K. Uz, and M. Vetterli, "Mutilresolution broadcast for digital HDTV using joint source/channel coding," *IEEE Journal on Selected Areas in Communications*, vol. 11, pp. 6–23, Jan. 1993.

[8] S. B. Z. Azami, O. Rioul, and P. Duhamel, "Performance bounds for joint source-channel coding of uniform memoryless sources using a binary decomposition," in *Proceedings of European Workshop on Emerging Techniques for Communication Terminals 1997*, pp. 259–263.

[9] B. Belzer, J. D. Villasenor, and B. Girod, "Joint source-channel coding of image with trellis coded quantization and convolutional codes," in *Proceedings of ICIP 1995*, vol. 2, pp. 85–88.

[10] G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, pp. 72–81, Jul. 1997.

[11] H. Man, F. Kossentini, and M. Smith, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, pp. 8–11, Aug. 1997.

[12] J. Lu, A. Nosratinia, and B. Aazhang, "Progressive source-channel coding of images over bursty error channels," in *Proceedings of ICIP 1998*, vol. 2, pp. 127–131.

[13] M. P. C. Fossorier, Z. Xiong, and K. Zeger, "Joint source-channel image coding for a power constrained noisy channel," in *Proceedings of ICIP 1998*, vol. 2, pp. 122–126.

[14] M. Brystrom and J. W. Modestino, "Combined source channel coding for transmission of video over a slow-fading Rician channel," in *Proceedings of ICIP 1998*, vol. 2, pp. 147–151.

[15] T.-H. Lan and A. H. Tewfik, "Power optimized mode selection for H.263 video coding and wireless communications," in *Proceedings of ICIP 1998*, vol. 2, pp. 113–117.

[16] H. Zheng and K. J. R. Liu, "Image and video transmission over wireless channel: A subband modulation approach," in *Proceedings of ICIP 1998*, vol. 2, pp. 132–136.

[17] Z. Xiong, B.-J. Kim, and W. A. Pearlman, "Progressive video coding for noisy channels," in *Proceedings of ICIP 1998*, vol. 1, pp. 334–337.

[18] S. Aramvith, I.-M. Pao, and M.-T. Sun, "A rate-control scheme for video transport over wireless channels," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 11, pp. 569–580, May. 2001.

[19] S. Appadwedula, D. L. Jones, K. Ramchandran, and I. Konzintsev, "Joint source-channel matching for a wireless communications link," in *Proceedings of ICC 1998*, pp. 482–486.

[20] S. Appadwedula, D. L. Jones, K. Ramchandran, and I. Konzintsev, "Joint source-channel matching for a wireless communications link," in *Proceedings of DCC 1998*, p. 523.

[21] A. Murat Tekalp, *Digital Video Processing*. Beijing: Prentice Hall, 1998.

[22] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, pp. 4–21, Apr. 1991.

[23] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, 1996.

[24] D. Fronczak and D. Seltz, "Motion-JPEG and MPEG solutions for multimedia," in *Conference Record of WESCON 1995*, p. 738.

[25] S.-Y. Huang and J.-S. Wang, "A low-cost desktop videoconferencing codec: an adaptive motion-JPEG design," *IEEE Transaction on Consumer Electronics*, vol. 40, pp. 944–950, Nov. 1994.

[26] S. Okada, Y. Matsuda, T. Watanabe, and K. Kondo, "A single chip motion JPEG codec LSI," *IEEE Transaction on Consumer Electronics.*, vol. 43, pp. 418–422, Aug. 1997.

[27] H. Yamauchi, S. Okada, Y. Matsuda, T. Mori, T. Watanabe, S. Okada, A. Kobayashi, i. Ogura, and Y. Harada, "One-chip 15 frame/s mega-pixel real-time image processor," in *Proceedings of ISSCC 2001*, pp. 144–145.

[28] Y-W. Lei and M. Ouhyoung, "Software-baed motion JPEG with progressive refinement for computer animation," *IEEE Transaction on Consumer Electronics*, vol. 40, pp. 557–562, Aug. 1994.

[29] W. Zheng and Z. Xiao, "A novel video coding scheme with frequency-domain-based conditional frame replenishment algorithm," in *Proceedings of ICICS 1997*, vol. 1, pp. 274–278.

[30] B. G. Haskell, F. W. Mounts, and J. C. Candy, "Interframe coding of videotelephone pictures," *Proceedings of the IEEE*, vol. 60, pp. 792–800, 1972.

[31] B.-J Kim and W. A. Peralman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *Proceedings of DCC 1997*, pp. 251–260.

[32] G. Lin and Z. Liu, "3D wavelet video codec and its rate control in ATM network," in *Proceedings of IEEE International Symposium on Circuits and Systems 1999*, vol. 4, pp. 447–450.

[33] J. Karlekar and U. B. Desai, "SPIHT video coder," in *Proceedings of IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control 1998*, vol. 1, pp. 45–48.

[34] N. D. Doulamis, A. D. Doulamis, G. E. Konstantoulakis, and G. I. Stassinopoulos, "Efficient modeling of VBR mpeg-1 coded video sources," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 10, pp. 93–112, Feb. 2000.

[35] C-Y. Hsu and A. Ortega, "Rate control for robust video transmission over wireless channels," in *Proceedings of VCIP 1997*, vol. 3024, pp. 1200–1211.

[36] A. Ortega and M. Khansari, "Rate control for video coding over variable bit rate channels with applications to wireless transmission," in *Proceedings of ICIP 1995*, vol. 3, pp. 388–391.

[37] R. E. Blahut, *Digital Transmission of Information*. Reading, MA: Addison-Wesley, 1990.

[38] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Tranaction on Communications*, vol. 36, pp. 389–400, Apr. 1988.

[39] J. Hagenauer, N. Seshadri, and C-E. W .Sundberg, "The performance of rate-compatible punctured convolutional codes for digital mobile radio," *IEEE Transaction on Communications*, vol. 38, pp. 966–980, July 1990.

[40] J. Hagenauer and T. Stockhammer, "Channel coding and transmission aspects for wireless multimedia," *Proceedings of the IEEE*, vol. 87, pp. 1764–1777, 1999.

[41] D. E. Comer, *Internetworking with TCP/IP*. Upper Saddle River, NJ: Prentice Hall, 1995.

[42] A. S. Tanenbaum, *Computer Networks*. Upper Saddle River, NJ: Prentice Hall, 1994.

[43] J. R. Li, X. Gao, L. Qian, and V. Bharghavan, "Goodput control for heterogeneous data streams," in *Proceedings of the 10th International Workshop on Network and Operating System Supoort for Digital Audio and Video*, Jun. 2000, pp. 482–491.

[44] L. Qian, "Joint source-channel matching for wireless video transmission," M.S. thesis, University of Illinois at Urbana-Champaign, Jan. 1998.

[45] M. Khansari and M. Vetterli, "Layered transmission of signals over power-constrained wireless channels," in *Proceedings of ICIP 1995*, vol. 3, pp. 380–383.

[46] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Scalable video coding and transport over broad-band wireless networks," *Proceedings of the IEEE*, vol. 89, pp. 6–20, 2001.

[47] X. Jun and A. R. Barron, "Asymptotic minimax regret for data compression, gambling, and prediction," *IEEE Transaction on Information Theory*, vol. 46, pp. 431–445, March 2000.

[48] R. Rejaie, M. Handley, and D. Estrin, "Layered quality adaptation for Internet video streaming," *IEEE Transaction on Selected Areas in Communications*, vol. 18, pp. 2530–2543, Dec. 2000.

[49] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operational Research Society*, vol. 49, pp. 237–252, 1998.

[50] M. Freytes, C. E. Rodriguez, and C. A. Marques, "Real-time H.263+ video transmission on 802.11 wireless LANs," in *Proceedings of International Conference on Information Technology: Coding and Computing 2001*, pp. 125–129.

[51] J. Cai and C. W. Chen, "FEC-based video streaming over packet loss networks with pre-interleaving," in *Proceedings of International Conference on Information Technology: Coding and Computing 2001*, pp. 125–129.

[52] B. J. Vickers, C. Albuquerque, and T. Suda, "Source-adaptive multilayered multicast algorithms for real-time video distribution," *IEEE/ACM Transaction on Networking*, vol. 8, pp. 720–733, Dec. 2000.

[53] R. Kim, B. Roh, and J. Kim, "Bandwidth regegotiation with traffic smoothing and joint rate control for VBR mpeg video over ATM," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 10, pp. 693–703, Aug. 2000.

[54] D. Wu, Y. T. Hou, W. Zhu, H. Lee, T. Chiang, Y.-Q. Zhang, and H. J. Chao, "On end-to-end architecture for transporting MPEG-4 video over the internet," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 10, pp. 932–941, Sept. 2000.

[55] J. W. Kim, Y.-G. Kim, H. S. Song, T.-Y. Kuo, Y. J. Chung, and C.-C. J. Kuo, "TCP-friendly Internet video streaming employing variable frame-rate encoding and interpolation," *IEEE Transaction on Circuits and Systems for Video Technology.*, vol. 10, pp. 1164–1177, Oct. 2000.

[56] H. Liu, N. Ansari, and Y. Q. Shi, "On-line dynamic bandwidth allocation for VBR video tranmission," in *Proceedings of IEEE International Conference on Information Technology: Coding and Computing 2001*, pp. 354–358.

[57] W. Zhu, Q. Zhang, and Y.-Q. Zhang, "Network-adaptive rate control with unequal loss protection for scalable video over Internet," in *Proceedings of IEEE International Symposium on Circuits and Systems 2001*, vol. 5, pp. 109–112.

[58] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transaction on Networking*, vol. 2, pp. 1–15, 1994.

[59] H. Wang and N. Moayeri, "Finite-state Markov channel-A useful model for radio communication channels," *IEEE Transaction on Commmunications*, vol. 44, pp. 163–171, Feb. 1995.

[60] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, "A multifractal wavelet model with application to network traffic," *IEEE Transaction on Information Theory*, vol. 45, pp. 992–1018, Apr. 1999.

[61] F. Kelly, "Mathematical modelling of the internet," in *Proceedings of the 4th International Congress on Industrial and Applied Mathematics*, July 1999.

[62] Y. S. Sun, F. M. Tsou, M. C. Chen, and Z. Tsai, "A TCP-friendly congestion control scheme for real-time packet video using prediction," in *Proceedings of GLOBECOM 1999*, vol. 3, pp. 1818–1822.

[63] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transaction*, vol. 7, no. 4, pp. 458–472, Aug 1999.

[64] M. Gerla, W. Weng, and R. L. Cigno, "Bandwidth feedback control of TCP and real time sources in the Internet," in *Proceedings of IEEE GLOBECOM 2000*, vol. 1, pp. 561–565.

[65] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, pp. 28–37, May-June 2001.

[66] Y. R. Yang, S. K. Min, and S. S. Larn, "Transient behaviors of TCP-friendly congestion control protocols," in *Proceedings of INFOCOM 2001*, vol. 3, pp. 1716–1725.

[67] M. Zorzi, R. R. Rao, and L. B. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Transaction on Vehicle Technology*, vol. 46, pp. 445–455, May. 1997.

[68] H. Liu and M. E. Zarki, "Performance of H.263 video transmission over wireless channels using hybrid ARQ," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 9, pp. 567–571, Dec. 1997.

[69] I. Rhee and S. R. Joshi, "Error recovery for interactive video tranmission over the internet," *IEEE Transaction on Selected Areas in Communications*, vol. 18, pp. 1033–1049, Jun. 2000.

[70] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Transaction on Selected Areas in Communications*, vol. 15, pp. 983–1001, 1997 1997.

[71] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra, "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video," in *Proceedings of DCC 2000*, pp. 440–449.

[72] R. Fu, L. B. Sung, and A. Gupta, "Scalable layered MPEG-2 video multicast architecture," *IEEE Transaction on Consumer Electronics*, vol. 47, pp. 55–62, Feb. 2001.

[73] J. Padley, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of SIGCOMM 1998*.

[74] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Network-adaptive rate control with tcp-friendly protocol for multiple video objects," in *Proceedings of IEEE International Conference on Multimedia and Expo 2000*, vol. 2, pp. 1055–1058.

[75] L-J. Lin, A. Ortega, and C.-C Jay Kuo, "Cubic spline approximation of rate and distortion functions for MPEG video," in *Proceedings of IST/SPIE, Digital Video Compression: Algorithms and Technologies 1996*, vol. 2668, pp. 169–180.

[76] G. Sudhir, J. C. M. Lee, and A. K. Jain, "Automatic classification of tennis video for high-level content-based retrieval," in *Proceedings of the 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, pp. 81–90.

[77] Y. Wang, J. Huang, Z. Liu, and T. Chen, "Multimedia content classification using motion and audio information," in *Proceedings of the 1997 IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 1488–1491.

[78] C-Y. Hsu and A. Ortega, "A Lagrangian optimization approach to rate control for delay-constrained video transmission over burst-error channels," in *Proceedings of ICASSP 1998*, vol. 5, pp. 2989–2992.

[79] L-J. Lin, A. Ortega, and C.-C. Jay Kuo, "A gradient-based rate control algorithm with applications to MPEG video," in *Proceedings of ICIP 1995*, vol. 3, pp. 392–395.

[80] L-J. Lin and A. Ortega, "Bit-rate control using piecewise approximated rate-distortion characteristics," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 3, pp. 665–676, Feb. 1998.

[81] V. F. Demyanov and V. N. Malozemov, *Introduction to Minimax*. NY: Dover Publications, 1990.

VITA

Leiming Qian was born in Lishui, Zhejiang province, People's Republic of China. He received the B.E. degree in mechanical engineering from Tsinghua University, Beijing, P. R. C, in 1996. He received the M.S. degree in electrical engineering from the University of Illinois at Urbana-Champaign, in 1999.

Since 1996, he has been working as a research assistant at the University of Illinois at Urbana-Champaign. His current research interests include joint source-channel coding, image and video compression techniques, and network video streaming protocols.