EURASIP Journal on Applied Signal Processing

Multimedia over IP and Wireless Networks

Guest Editors: Zixiang Xiong, Mihaela van der Schaar, Jie Chen, Eckehard Steinbach, C.-C. Jay Kuo, and Ming-Ting Sun

EURASIP Journal on Applied Signal Processing

Multimedia over IP and Wireless Networks

EURASIP Journal on Applied Signal Processing

Multimedia over IP and Wireless Networks

Guest Editors: Zixiang Xiong, Mihaela van der Schaar, Jie Chen, Eckehard Steinbach, C.-C. Jay Kuo, and Ming-Ting Sun

Contents

Medusa: A Novel Stream-Scheduling Scheme for Parallel Video Servers, Hai Jin, Dafu Deng, and Liping Pang
Volume 2004 (2004), Issue 2, Pages 317-329

EURASIP Journal on Applied Signal Processing 2004:2, 155–157

Editorial

Zixiang Xiong
Department of Electrical Engineering, Texas A&M University, College Station, TX 77843, USA
Email: zx@ee.tamu.edu

Mihaela van der Schaar
Department of Electrical and Computer Engineering, University of California, Davis, CA 95616-5294, USA
Email: mvanderschaar@ece.ucdavis.edu

Jie Chen
Division of Engineering, Brown University, Providence, RI 02912-9104, USA
Email: jie_chen@brown.edu

Eckehard Steinbach
Institute of Communication Networks, Munich University of Technology, 80290 Munich, Germany
Email: eckehard.steinbach@tum.de

C.-C. Jay Kuo
Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089, USA
Email: cckuo@sipi.usc.edu

Ming-Ting Sun
Department of Electrical Engineering, University of Washington, Seattle, WA 98195-2500, USA
Email: sun@ee.washigton.edu

Multimedia—an integrated and interactive presentation of speech, audio, video, graphics, and text—has become a major driving force behind a multitude of applications. Increasingly, multimedia content is being accessed by a large number of diverse users and clients at anytime, and from anywhere, across various communication channels such as the Internet and wireless networks. As mobile cellular and wireless LAN networks are evolving to carry multimedia data, an all-IP-based system akin to the Internet is likely to be employed due to its cost efficiency, improved reliability, allowance of easy implementation of new services, independence of control and transport, and importantly, easy integration of multiple networks.

However, reliable transmission of multimedia over such an integrated IP-based network poses many challenges. This is not just due to the inherently lower transmission rates provided by these networks as compared with traditional delivery networks (e.g., ATM, cable networks, satellite), but also due to associated problems such as congestion, competing traffic, fading, interference, and mobility, all of which lead to varying transmission capacity and losses.

Consequently, to achieve a high level of acceptability and proliferation of networked multimedia, a solution for reliable and efficient transmission over IP and wireless networks is required. Several key requirements need to be satisfied.

(1) *Easy adaptability to rate variations* since the available transmission capacity may vary due to interference, overlapping wireless LANs, competing traffic, mobility, multipath fading, and so forth.
(2) *Robustness to data losses* since depending on the channel condition, partial data losses may occur.
(3) *Support for device scalability and user preferences* since various clients may be connected at different data rates and request transmissions that are optimized for their respective connections and capabilities.
(4) *Limited complexity implementations* for mobile wireless devices.

(5) *Adaptation to the quality-of-service* (*QoS*) provided by the network.
(6) *Efficient end-to-end transmission* over different networks exhibiting various characteristics and QoS guarantees.

To address the above-mentioned requirements, innovative solutions are needed for adaptive and error-resilient multimedia compression, error control, error protection and concealment, multimedia streaming architectures, channel models and channel estimation, packetization and scheduling, and so forth. Such solutions can best be developed by a combination of theory, tools, and methods from the fields of networking, signal processing, and computer engineering. This integrated and cross-disciplinary approach has led to the advent of a new research wave in compression, joint source-channel coding, and network-adaptive media delivery, and has motivated the emergence of novel compression standards, transmission protocols, and networking solutions.

Recently, both the academic and industrial communities have realized the potential of such integrated solutions for multimedia applications. Consequently, multimedia networking is evolving as one of the most active research areas. Despite the significant research efforts in this area, numerous problems related to the optimal design of source coding schemes aimed at transmission over a variety of networks, joint source-channel coding trade-offs, and flexible multimedia architectures remain open.

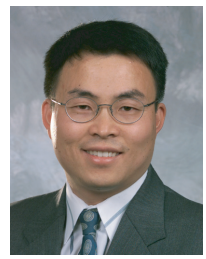
This special issue is an attempt to cover a wide range of topics under the broad multimedia networking umbrella by publishing twelve papers reporting on recent results in the above-mentioned research areas. The papers in this special issue correspond to advances in five different areas of multimedia networking:

(i) layered coding and transmission,
(ii) cost-effective and complexity-scalable implementations,
(iii) efficient end-to-end transmission using proxies,
(iv) quality of service,
(v) mechanisms for robust coding and transmission.

In the first area, Viéron et al., T. P.-C. Chen and T. Chen, Wu et al., and Thie and Taubman dedicate four papers, respectively, to robust video transmission using layered coding, covering various aspects such as joint source-channel coding, rate-shaping, and efficient streaming strategies. In the second area, Saponara et al. and Mietens et al. consider cost-effective and complexity-scalable implementations of the different video compression standards employed for multimedia communication applications. In the third area, Pei and Modestino, and Radha et al. consider the use of proxies for improving the video quality when transmitted over multiple-hop wireless or wired networks exhibiting different channel characteristics. In the fourth area, Song and Lee consider the effective mechanisms for QoS using renegotiating schemes for streaming video. In the fifth area, Taal et al., Song and Liu, and Jin et al. consider different mechanisms for robust video coding and transmission, such as source-channel rate allocation schemes, novel scheduling strategies for video distribution using parallel servers, and optimization of error-resilient video transmission using behavior models.

As this special issue illustrates, academic and industrial research in multimedia networking is becoming increasingly vibrant, and the field continues to pose new challenges that will require innovative approaches. Potential solutions will need to cross the boundaries between the fields of signal processing, networking, and computer engineering, and we believe that such cross-fertilization is likely to catalyze many interesting and relevant new research topics and applications.

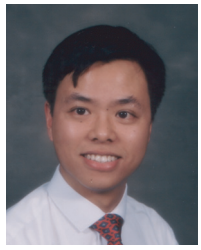
Zixiang Xiong
Mihaela van der Schaar
Jie Chen
Eckehard Steinbach
C.-C. Jay Kuo
Ming-Ting Sun



Zixiang Xiong received his Ph.D. degree in electrical engineering in 1996 from the University of Illinois at Urbana-Champaign. From 1997 to 1999, he was with the University of Hawaii. Since 1999, he has been with the Department of Electrical Engineering at Texas A&M University, where he is an Associate Professor. He spent the summers of 1998 and 1999 at Microsoft Research, Redmond, Wash and the summers of 2000 and 2001 at Microsoft Research in Beijing. His current research interests are distributed source coding, joint source-channel coding, and genomic signal processing. Dr. Xiong received a National Science Foundation (NSF) Career Award in 1999, an Army Research Office (ARO) Young Investigator Award in 2000, and an Office of Naval Research (ONR) Young Investigator Award in 2001. He also received Faculty Fellow Awards in 2001, 2002, and 2003 from Texas A&M University. He is currently an Associate Editor for the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Transactions on Signal Processing, and the IEEE Transactions on Image Processing.

Mihaela van der Schaar is currently an Assistant Professor in the Electrical and Computer Engineering Department at the University of California, Davis. She received her Ph.D. degree in electrical engineering from Eindhoven University of Technology, the Netherlands. Between 1996 and June 2003, she was a Senior Member Research Staff at Philips Research in the Netherlands and USA. In 1998, she worked in the Wireless Communications and Networking Department. From January to September 2003, she was also an Adjunct Assistant Professor at Columbia University. In 1999, she become an active participant to the MPEG-4 standard, contributing to the scalable video coding activities. She is currently chairing the MPEG Ad-hoc group on Scalable Video Coding, and is also cochairing the Ad-hoc group on Multimedia Test Bed. Her research interests include multimedia coding, processing, networking, and architectures. She has authored more than 70 book chapters, and conference and journal

papers and holds 9 patents and several more pending. She was also elected as a member of the Technical Committee on Multimedia Signal Processing of the IEEE Signal Processing Society and is an Associate Editor of IEEE Transactions on Multimedia and an Associate Editor of Optical Engineering.



Jie Chen received his M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park. He is currently an Assistant Professor at Brown University in the Division of Engineering, and the head of Brown BINARY lab. From 2000 to 2002, he has worked as a Principal System Engineer of two startup companies, first Lucent Digital Radio, then cofounded Flarion Technology. Dr. Chen's research interests include multimedia communication, nano-scale device modeling, and genomic signal processing. He has received NSF Award, Division Award from Bell Labs, and Student Paper Award. He has been invited as the speaker in different conferences and workshops. Since 1997, Dr. Chen has authored or coauthored 46 scientific papers in refereed journals and conference proceedings—35 as the first author. He first-authored the book *Design of Digital Video Coding Systems: A Complete Compressed Domain Approach* (New York: Marcel Dekker 2001); and coedited another textbook, *Genomic Signal Processing and Statistics* (EURASIP Book Series, 2004). He has invented or coinvented seven US patents. Currently, he is the Associate Editor of *IEEE Signal Processing Magazine, IEEE Transactions on Multimedia, and EURASIP Journal on Applied Signal Processing.*

Eckehard Steinbach studied electrical engineering at the University of Karlsruhe, Germany, the University of Essex, UK, and ESIEE in Paris. From 1994 to 2000, he was a member of the research staff of the Image Communication Group at the University of Erlangen-Nuremberg, Germany, where he received the Engineering Doctorate in 1999. From February 2000 to December 2001, he was a Postdoctoral Fellow at the Information Systems Laboratory, Stanford University. In February 2002, he joined the Department of Electrical Engineering and Information Technology, Technische Universität München, Germany, where he is currently an Associate Professor of media technology. Dr. Steinbach served as a Conference Cochair of SPIE Visual Communications and Image Processing (VCIP '01) in San Jose, California, in 2001. He also served as a Conference Cochair of Vision, Modeling, and Visualization (VMV '03) held in Munich in November 2003. His current research interests are in the area of networked multimedia systems.

C.-C. Jay Kuo received his B.S. degree from the National Taiwan University, Taipei, in 1980 and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively, all in electrical engineering. Since January 1989, he has been with the Department of Electrical Engineering Systems at the University of Southern California. His research interests are in the areas of digital signal and image processing, audio and video coding, multimedia communication technologies and delivery protocols, and embedded system design. Kuo is a Fellow of IEEE and SPIE and a Member of ACM. He is Editor-in-Chief of the Journal of Visual Communication and Image Representation, Associate Editor of IEEE Transactions on Speech and Audio Processing, and Editor of the EURASIP Journal on Applied Signal Processing. He is also in the Editorial Board of the IEEE Signal Processing Magazine. He received the National Science Foundation Young Investigator Award (NYI) and Presidential Faculty Fellow (PFF) Award in 1992 and 1993, respectively. He has guided about 52 students to their Ph.D. degrees and supervised 9 Postdoctoral Research Fellows. He is a coauthor of six books and more than 600 technical publications in international conferences and journals.

Ming-Ting Sun received the B.S. degree from National Taiwan University in 1976, the M.S. degree from University of Texas at Arlington in 1981, and the Ph.D. degree from University of California, Los Angeles in 1985, all in electrical engineering. Dr. Sun joined the University of Washington in August 1996 where he is now a Professor. His research interests include video coding and networking, multimedia technologies, and VLSI for signal processing. Dr. Sun has been awarded 8 patents and has published more than 140 technical papers in journals and conferences. He has authored or coauthored 10 book chapters in the area of video technology, and has coedited a book on compressed video over networks. He has served in various leadership positions including the Chair of the IEEE CAS Standards Committee, the Editor-in-Chief of IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), General Cochair of Visual Communication and Image Processing, and the Editor-in-Chief of IEEE Transactions on Multimedia. Dr. Sun has received many awards including the Award of Excellence from Bellcore, the *TCSVT* Best Paper Award, and the Golden Jubilee Medal from the IEEE CAS Society. Dr. Sun is a Fellow of IEEE.

EURASIP Journal on Applied Signal Processing 2004:2, 158–175

Source and Channel Adaptive Rate Control for Multicast Layered Video Transmission Based on a Clustering Algorithm

Jérôme Viéron
Thomson multimedia R&D, 1 avenue Bellefontaine - CS 17616, 35576 Cesson-Sévigné, France
Email: jerome.vieron@inria.fr

Thierry Turletti
INRIA, 2004 route des Lucioles - BP 93, 06902 Sophia Antipolis Cedex, France
Email: thierry.turletti@inria.fr

Kavé Salamatian
Laboratoire d'Informatique de Paris 6 (LIP6), 8 rue du Capitaine Scott, 75015 Paris, France
Email: kave.salamatian@inria.fr

Christine Guillemot
INRIA, Campus de Beaulieu, 35042 Rennes Cedex, France
Email: christine.guillemot@inria.fr

Received 24 October 2002; Revised 8 July 2003

This paper introduces source-channel adaptive rate control (SARC), a new congestion control algorithm for layered video transmission in large multicast groups. In order to solve the well-known feedback implosion problem in large multicast groups, we first present a mechanism for filtering RTCP receiver reports sent from receivers to the whole session. The proposed filtering mechanism provides a classification of receivers according to a predefined similarity measure. An end-to-end source and FEC rate control based on this distributed feedback aggregation mechanism coupled with a video layered coding system is then described. The number of layers, their rate, and their levels of protection are adapted dynamically to aggregated feedbacks. The algorithms have been validated with the NS2 network simulator.

Keywords and phrases: multicast, congestion control, layered video, aggregation, FGS.

1. INTRODUCTION

Transmission of multimedia flows over multicast channels is confronted with the receivers heterogeneity problem. In a multicast topology (multicast delivery tree in the $1 \rightarrow N$ case, acyclic graph in the $M \rightarrow N$ case), network conditions such as loss rate (LR) and queueing delays are not homogeneous in the general case. Rather, there may be local congestions affecting downstream delivery of the video stream in some branches of the topology. Hence, the different receivers are connected to the source via paths with varying delays, loss, and bandwidth characteristics. Due to this potential heterogeneity, dynamic adaptation of multimedia flows over multicast channels, for optimized quality-of-service (QoS) of multimedia sessions, faces challenging problems. The adaptation of source and transmission parameters to the network state often relies on the usage of feedback mechanisms. However, the use of feedback schemes in large multicast trees faces the potential problem of feedback implosion. This paper introduces source-channel adaptive rate control (SARC), a new congestion control algorithm for layered video transmission in large multicast groups. The first issue addressed here is therefore the problem of aggregating heterogeneous reports into a consistent view of the communication state. The second issue concerns the design of a source rate control mechanism that would allow a receiver to receive the source signal with a quality commensurate with the bandwidth and loss capacity of the path leading to it.

Layered transmission has been proposed to cope with receivers heterogeneity [1, 2, 3]. In this approach, the source is represented using a base layer (BL) and several successive enhancement layers (EL) refining the quality of the source reconstruction. Each layer is transmitted over a separate multicast group, and receivers decide the number of groups to join (or leave) according to the quality of their reception. At the other side, the sender can decide the optimal number of layers and the encoding rate of each layer according to the feedback sent by all receivers. A variety of multicast schemes making use of layered coding for audio and video communication have been proposed, some of which rely on a multicast feedback scheme [3, 4]. Despite rate adaptation to the network state, applications have to face the remaining packet losses. Error control schemes using *forward error correction* (FEC) strongly reduce the impact of packet losses [5, 6, 7]. In these schemes, redundant information are sent along with the original information so that the lost data (or at least part of it) can be recovered from the redundant information. Clearly, sending redundancy increases the probability of recovering the lost packets, but it also increases the bandwidth requirements, and thus the LR of the multimedia stream. Therefore, it is essential to couple the FEC scheme to the rate control scheme in order to jointly determine the transmission parameters (redundancy level, source coding rate, type of FEC scheme, etc.) as a function of the state of the multicast channel, to achieve the best subjective quality at receivers. For such adaptive mechanisms, it is important to have simple channel models that can be estimated in an online manner.

The sender, in order to adapt the transmission parameters to the network state, does not need reports of each receiver in the multicast group. It rather needs a partition of the receivers into homogeneous classes. Each layer of the source can then be adapted to the characteristics of one class or of a group of classes. Each class represents a group of homogeneous receivers according to discriminative variables related to the received signal quality. The clustering mechanism used here follows the above principles. A classification of receiver reports (RRs) is performed by aggregation agents (AAs) organized into a hierarchy of local regions. The approach assumes the presence of AAs at strategic positions within the network. The AAs classify receivers according to similar reception behaviors and filter correspondingly the (real-time transport control protocol) RTCP RRs. By classifying receivers, this mechanism solves the feedback implosion problem and at the same time provides the sender with a compressed representation of the receivers.

In the experiments reported in this paper, we consider two pairs of discriminative variables in the clustering process: the first one constituted of the LR and the *goodput* and the second constituted of the LR and the throughput of a conformant TCP (transport control protocol) connection under similar loss and round-trip time (RTT) conditions. We show approaches in which receivers rate requests are only based on the goodput measure risk leading to a severe subutilization of the network resources. To use a TCP throughput model, receivers have to estimate their RTT to the source first. In order to do so, we use the algorithm described in [4] jointly with a new application-defined RTCP packet, called *probe RTT*.

This distributed feedback aggregation mechanism is coupled with a video fine-grain scalable (FGS) layered coding system to adapt dynamically the number of layers, the rate of each layer, and its level of protection. Notice that the aggregation mechanism that has to be supported by the network nodes remains generic and can be used for any type of media. The optimization is performed by the sender and takes into account both the network aggregated state as well as the rate-distortion characteristics of the source. The latter allows to optimize the quality perceived by each receiver in the multicast tree.

The remainder of this paper is organized as follows. Section 2 provides an overview of related research on multicast rate and congestion control. Section 3 sets the main lines of SARC, our new hybrid sender/receiver driven rate control based on a clustering algorithm. The protocol functions to be supported by the receivers and the receiver clustering mechanism governing the feedback aggregation are described, respectively, in Sections 4 and 5. Section 6 describes the multilayer source and channel rate control and the multi-layered MPEG-4 FGS source encoder [8, 9] that have been used in the experiments. Finally, experimental results obtained with the NS2 network simulator with various discriminative clustering variables (goodput, TCP-compatible throughput), including the additional usage of FEC are discussed in Section 7.

2. RELATED WORK

Related work in this area focuses on error, rate, and congestion control in multicast for multimedia applications. Layered coding is often proposed as a solution for rate control in video multicast applications over the Internet. Several approaches—sender-driven [10], receiver-driven [11, 12], or hybrid schemes [3, 13, 14]—have been proposed to address the problem of rate control in a multicast transmission. Receiver-driven approaches consist in multicasting different layers of video using different multicast addresses and let the receivers decide which multicast group(s) to subscribe to. RLM (receiver-driven layered multicast) [11] and RLC (radio link control) [12] are two well-known receiver-driven layered multicast congestion control protocols. However, they both suffer from pathological behaviors such as transient periods of congestion, instability, and periodic losses. These problems mainly come from the bandwidth inference mechanism used [15]. For example, RLM uses *join experiments* that can create additional traffic congestion during transition periods corresponding to the latency for pruning a branch of the multicast tree. RLC [12] is a TCP-compatible version of RLM, based on the generation of periodic bursts that are used for bandwidth inference on synchronization points indicating when a receiver can join a layer. Both the synchronization points and the periodic bursts can lead to periodic

congestion and periodic losses [15]. PLM (Packet-pair layered multicast) [16] is a more recent layered multicast congestion control protocol, based on the generation of packet pairs to infer the available bandwidth. PLM does not suffer from the same pathological behaviors as RLM and RLC but requires a fair queuing network.

Bhattacharya et al. [17] present a general framework for the analysis of additive increase multiplicative decrease (AIMD) multicast congestion control protocols. This paper shows that because of the so-called "path loss multiplicity problem," unclever use of congestion information sent by receivers to 1 sender may lead to severe degradation and lack of fairness. This paper formalizes the multicast congestion control mechanism in two components: the loss indication filter (LIF) and the rate adjustement algorithm. Our paper presents an implementation that minimises the loss multiplicity problem by using an LIF which is implemented by a clustering mechanism (Section 5.2) and a rate adjustement algorithm following the algorithm described in Sections 4 and 6.

TFMCC [18] is an equation-based multicast congestion control mechanism that extends the TCP-friendly TFRC [19] protocol from the unicast to the multicast domain. TFMCC uses a scalable RTT measurement and a feedback suppression mechanism. However, since it is a single-rate congestion control scheme, it cannot handle heterogeneous receivers and adapts its sending rate to the current limiting receiver.

FLID-DL [20] is a multirate congestion control algorithm for layered multicast sessions. It mitigates the negative impact of long Internet group management protocol (IGMP) leave latencies and eliminates the need for probe intervals used in RLC. However, the amount of IGMP and PIM-SM (protocol independent multicast-sparse mode) control traffic generated by each receiver is prohibitive. WEBRC [21] is a new equation-based rate control algorithm that has been recently proposed. It solves the main drawbacks of FLID-DL using an innovative way to transmit data in waves. However, WEBRC, such as FLID-DL, is intended for reliable download applications and possibly streaming applications but cannot be used to transmit real-time hierarchical flows such as H.263+ or MPEG-4.

A source adaptive multilayered multicast (SAMM) algorithm based on feedback packets containing information on the estimated bandwidth (EB) available on the path from the source is described in [3]. Feedback mergers are assumed to be deployed in the network nodes to avoid feedback implosion. A mechanism based on *partial suppression* of feedbacks is proposed in [4]. This approach avoids the deployment of aggregation mechanisms in the network nodes, but on the other hand, the partial feedback suppression will likely induce a flat distribution of the requested rates.

MLDA [13] is a TCP-compatible congestion control scheme in which, as in the scheme we propose, senders can adjust their transmission rate according to feedback information generated by receivers. However, MLDA does not provide a way to adapt the FEC rate in the different layers according to the packet loss observed at receivers. Since the feedback only includes TCP-compatible rates, MLDA does not need feedback aggregation mechanisms and uses exponentially distributed timers and a partial suppression mechanism to prevent feedback implosion. However, when the receivers are very heterogeneous, the number of requested rates (in the worst case on a continuous scale) can potentially lead to a feedback implosion. Moreover, the partial suppression algorithm does not allow quantifying the number of receivers requesting a given rate in order to estimate how representative this rate is.

In [14], a rate-based congestion and loss control mechanism for multicast layered video transmission is described. The strategy relies on a mechanism that aggregates feedback information in the networks nodes. However, in contrast with SAMM, the optimization is not performed in the nodes. Source and channel FEC rates in the different layers are chosen among a set of requested rates in order to maximize the overall peak signal-to-noise ratio (PSNR) seen by all the receivers. Receivers are classified according to their available bandwidth, and for each class of rate, two types of information are delivered to the sender: the number of receivers represented by this class and an average LR computed over all those receivers. It is supposed here that receivers with similar bandwidths have similar LRs, which may not always be the case. In this paper, we solve this problem using a distributed clustering mechanism.

Clustering approaches have been already considered separately in [22, 23]. In [22], a centralized classification approach based on k-means clustering is applied on a quality of reception parameter. This quality of reception parameter is derived, based on the feedback of receivers consisting of reports including the available bandwidth and packet loss. The main difference, compared with our approach, is that in our case, the classification is made in a distributed fashion. Hence, receivers with similar bandwidths but with different LRs are not classified within the same class. Therefore, with more accurate clusters, a better adaptation of the error control process at the source level is possible. The global optimization performed is different and leads to improved performances. Moreover, [22] uses the RTCP filtering mechanism proposed in the RTP (real-time transport protocol) standard, that is, they adapt the RTCP sending rate according to the number of receivers. However, when the number of receivers is large, it is not possible to get a precise snapshot of quality observed by receivers.

3. PROTOCOL OVERVIEW

This section gives an overview of the SARC protocol proposed in this paper. Its design relies on a feedback tree structure, where the receivers are organized into a tree hierarchy, and internal nodes aggregate feedbacks.

At the beginning of the session, the sender announces the range of rates (i.e., a rate interval $[R_{\min}, R_{\max}]$) estimated from the average rate-distortion characteristics of the source. The value $R_{\min}$ corresponds to the bit rate under which the

received quality would not be acceptable, whereas $R_{\max}$ corresponds to the rate above, under which there is no significant improvement of the visual quality. This information is transmitted to the receivers at the start of the session. The interval $[R_{\min}, R_{\max}]$ is then divided into subintervals in order to only allow relevant values for layers rates. This quantization avoids having nonquality discriminative layers.

After this initialization, the multicast layered rate control process can start. The latter assumes that the time is divided into feedback rounds. A feedback round comprises four major steps.

(i) At the beginning of each round, the source announces the number of layers and their respective rates via RTCP sender reports (SRs). Each source layer is transmitted to an Internet protocol (IP) multicast group.

(ii) Each receiver measures network parameters and estimates the bandwidth available on the path leading to it. The EB and the layer rates will trigger subscriptions or unsubscriptions to/from the layers. EB and LRs are then conveyed to the sender via RTCP RR.

(iii) AAs placed at strategic positions within the network classify receivers according to similar reception behaviors, that is, according to a measure of distance between the feedback parameter values. On the basis of this clustering, these agents proceed with the aggregation of the feedback parameters, providing a representation of homogeneous clusters.

(iv) The source then proceeds with a dynamic adaptation of the number of layers and of their rates in order to maximize the quality perceived by the different clusters.

Sections 4, 5, and 6 describe in details each of the four steps.

4. PROTOCOL FUNCTIONS SUPPORTED BY THE RECEIVER

Two bandwidth estimation strategies have been considered: the first approach measures the goodput of the path and the second estimates the TCP-compatible bandwidth under similar conditions of LRs and delays. This section describes the functions supported by the receiver in order to measure the corresponding parameters and the multicast groups join and leave policy that has been retained. The bandwidth values estimated by the receivers are then conveyed to the sender via RTCP RRs augmented with dedicated fields.

4.1. Goodput-based estimation

A notion of goodput has been exploited in the SAMM algorithm described in [3]. Assuming the priority-based differentiated services for the different layers, the goodput is defined as the cumulated rate of the layers received without any loss. If a layer has suffered from losses, it will not be considered in the goodput estimation. The drawback of such a measure is that the EB will be highly dependent on the sending rates, hence it does not allow an accurate estimation of the link capacity. When no loss occurs, in order to best approach the link capacity, SAMM considers values higher than the goodput measured. Nevertheless, a LR of 0% is not realistic on the Internet. Experiments have shown that this notion of goodput in a best-effort network, in presence of cross traffic, leads to EBs decreasing towards zero during the sessions. Here, the goodput is defined instead as the rate received by the end system. A simple mechanism has been designed to try to approach the bottleneck rate of the link. If the LR is under a given threshold T_{loss}, the bandwidth value B_t estimated at time t is incremented as

$$B_t = B_{t-1} + \Delta, \tag{1}$$

where Δ represents a rate increment and B_{t-1} represents the last estimated value. Let g_t be the observed goodput value at time t. Thus, when the LR becomes higher than the threshold T_{loss}, B_t is set to g_t.

In the experiments we have taken $t_{\text{loss}} = 3\%$ and the Δ parameter increases similarly to the TCP increase, that is, of one packet per RTT.

4.2. TCP-compatible bandwidth estimation

The second strategy considered for estimating the bandwidth available on the path relies on the analytical model of TCP throughput [24], known also as the TCP-compatible rate control equation. Notice, however, that the application of the model in a multicast environment is not straightforward.

4.2.1. TCP throughput model

The average throughput of a TCP connection under given delay and loss conditions is given by [24]:

$$T = \frac{\text{MSS}}{\text{RTT}\sqrt{2p/3} + T_o \min\left(1, 3\sqrt{3p/8}\right) p\left(1 + 32p^2\right)}, \tag{2}$$

where p, RTT, MSS, and T_o represent, respectively, the congestion event rate [19], the round-trip time, the maximum segment size (i.e., maximum packet size), and the retransmit time out value of the TCP algorithm.

4.2.2. Parameters estimation

In order to be able to use the above analytical model, each receiver must estimate the RTT on its path. This is done using a new application-defined RTCP packet that we called probe RTT. To prevent feedback implosion, only leaf aggregators are allowed to send probe RTT packets to the source. In case receivers are not located in the same LAN of their leaf aggregator, they should add the RTT to their aggregator; this can be easily estimated locally and without generating undesirable extra traffic. The source periodically multicasts RTCP reports including the RTT computed (in milliseconds) for the latest probe RTT packets received along with the corresponding SSRCs. Then, each receiver can update its RTT estimation using the result sent for its leaf aggregator. The estimation of

the congestion event rate p is done as in [25] and the parameter MSS is set to 1000 bytes.

4.2.3. Singular receivers

In highly heterogeneous environments, under constraints of bounded numbers of clusters, the rate received by some end systems may strongly differ from their requests, hence from the TCP-compatible throughput value. The resulting excessively low values of congestion event rates lead in turn to overestimated bandwidth values, hence to unstability. In order to overcome this difficulty, the TCP-compatible throughput B_t at time t is estimated as

$$B_t = \min\left(T, \max\left(S_{\text{rate}} + T_{\text{rate}}, B_{t-1}\right)\right), \tag{3}$$

where S_{rate} is the rate subscribed to, T_{rate} is a threshold chosen so that the increase between two requests is limited (i.e., $T_{\text{rate}} = K \times \text{MSS}/\,\text{RTT}$ with K a constant), and B_{t-1} is the last estimated value of the TCP-compatible throughput. When the estimated throughput value T is not reliable, the history used in the estimation of LRs is reinitialized using the method described in [19]. We will see in the experimentation results that the above algorithm is still reactive and responsive to changes in network conditions.

4.2.4. Slow-start mechanism

The slow-start mechanism adopted here differs from the approaches described in [18, 19]. At the beginning of the session or when a new receiver joins the multicast transmission tree, the requested rate is set to $R_{\min}$. Then, after having a first estimation of RTT and p, T can be computed and the resulting requested rate B_t^{slow} is given by

$$B_t^{\text{slow}} = \max\left(T, g_t + K \times \frac{\text{MSS}}{\text{RTT}}\right), \tag{4}$$

where g_t is the observed goodput value at time t and K is the same constant as the one used in Section 4.2.3. The estimation given by (4) is used until we observe the first loss. After the first loss, the loss history is reinitialized taking g_t as the available bandwidth and proceeding with (3).

4.3. Join/leave policy

Each receiver estimates its available bandwidth B_t and joins or leaves layers accordingly. However, the leaving mechanism has to take into account the delay between the instant in which a feedback is sent and the instant in which the sender adapts the layer rates accordingly. Undesirable oscillations of subscription may occur if receivers decide to unsubscribe a layer as soon as the TCP-compatible throughput estimated is lower than the current rate subscribed to. It is essential to leave enough time for the source to adapt its sending rates, and only then decide to drop a layer if the request has not been satisfied. That is why in order to be still reactive, we have chosen a delay of $K \times$ RTT before leaving a layer except in the case where the LR becomes higher than a chosen acceptable bound T_{loss} (K is the same constant as the one used in Section 4.2.3). These coupled mechanisms permit avoiding a waste of bandwidth due to IGMP traffic.

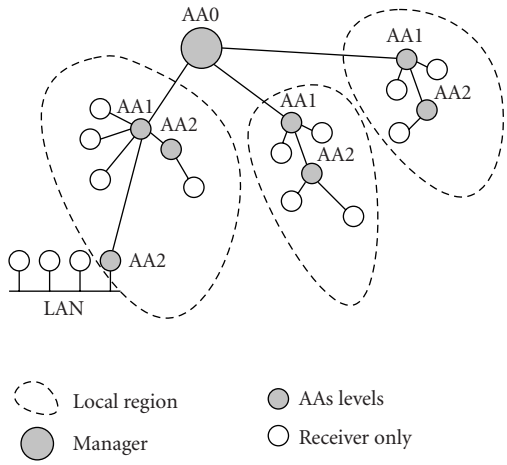


Figure 1: Multilevel hierarchy of aggregators.

4.4. Signalling protocol

The aggregated feedback information (i.e., EB and LR) are periodically conveyed towards the sender in RTCP RRs, using the RTCP report extension mechanism. The RRs are augmented with the following fields:

(i) EB: a 16-bit field which gives the value of the estimated bandwidth expressed in Kbps;

(ii) LR: a 16-bit field which gives the value of the real loss rate;

(iii) NB: a 16-bit field which gives the number of clients requesting this rate (i.e., EB). This value is set to one by the receiver.

5. AGGREGATED FEEDBACK USING DISTRIBUTED CLUSTERING

Multicast transmission has been reported to exhibit strong spatial correlations [26]. A classification algorithm can take advantage of this spatial correlation to cluster similar reception behaviors into homogeneous classes. In this way, the amount of feedback required to figure out the state of receivers can be significantly reduced. This will also help in bypassing loss path multiplicity problem explained in [17] by filtering out the receivers' report of losses. In our scheme, receivers are grouped into a hierarchy of local regions (see Figure 1). Each region contains an aggregator that receives feedback, performs some aggregation statistics, and send them in point-to-point to the higher level aggregator (*merger*). The root of the aggregator tree hierarchy (called the *manager*) is based at the sender and receives the overall aggregated reports.

This architecture has a slight modification compared to the generic RTP architecture. Similar to the PIM-SM context, RRs are not sent in multicast to the whole session, but are sent in point-to-point to a higher level aggregator. As these RTCP feedbacks are local to an aggregator region and will not cross the overall multicast tree, they may be set to be more frequent without breaking the 5% of the overall traffic constraint specified by the RTP standard.

5.1. Aggregators organization within the network

AAs must be set up at strategic positions within the network in order to minimize the bandwidth overhead of RTCP RRs. Several approaches have been proposed to organize receivers in a multicast session to make scalable reliable multicast protocols [27]. We have chosen a multilevel hierarchical approach such as that described in the RMTP [28] protocol in which receivers are also grouped into a hierarchy of local regions. However, in our approach, there are no designated receivers: all receivers send their feedback to their associated aggregator.

The root of the aggregator tree hierarchy (called the manager) is based at the sender and receives the overall summary reports. The maximal allowed height of the hierarchical tree is set to 3 as recommended in [29]. In our approach, the overall summary report is a classification containing the number of receivers in each class and the mean behaviour of the class. The mechanism of aggregation is described in Section 5.2.

In our experiments, aggregators are manually set up within the network. However, if extra router functionalities are available, several approaches can be used to automatically launch aggregators within the network. For example, we can implement the *aggregator* function using a *custom concast* [30]. *Concast* addresses are the opposite of multicast addresses, that is, they represent groups of senders instead of groups of receivers. So, a concast datagram contains a multicast group source address and a unicast destination address. With such a scheme, all receivers send their RRs feedback packets using the RTCP source group address to the sender's unicast address, and only one aggregated packet is delivered to the sender. The custom concast signaling interface allows the application to provide the network with the description of the merging algorithm function.

5.2. Clustering mechanism

The clustering mechanism is aimed towards taking advantage of the spatial and temporal correlation between the receiver's state of reception. Spatial correlation means that there is redundancy between reception behavior of neighbor receivers. This redundancy can be removed by compression methods. This largely reduces the amount of data required for representing feedback data sent by receivers. The compression is achieved by clustering similar (by a predefined similarity measure) reception behaviors into homogeneous classes. In this case, the clustering can be viewed as a vector quantization [31] that constructs a compact representation of the receivers as a classification of receivers issuing similar RRs. Moreover, for sender-based multicast regulation, only a classification of receivers is sufficient to apply adaptation decisions.

The clustering mechanism can also take advantage of time redundancy. For this purpose, classification of receivers should integrate the recent history of receivers as well as the actual RRs. Different reception states experienced by receivers during past periods are treated as reports of different and heterogeneous receivers. By this way, temporal variation of the quality of a receiver reception are integrated in the classification. A receiver that observes temporal variation may change its class during time.

In a stationary context, the classification would converge to a stable distribution. This stationary distribution will be a function of the spatial as well as the temporal dependencies. However, since over large time scales, the stationary hypothesis cannot be always validated, a procedure should be added to track variation of the multicast channel and adapt the classification to it. This procedure can follow a classical exponential weighting that drive the clustering mechanism to forget about far past-time reports. In this weighting mechanism, the weight of clusters is multiplied by a factor ($\gamma < 1$) at the end of each reporting round, and clusters with weight below a threshold are removed.

Before describing the classification algorithm, several concepts should be introduced. First, we should choose the discriminative characteristic and the similarity (or dissimilarity) measure needed to detect similar reception behavior.

5.2.1. Discriminative network characteristics

In the system presented in this paper, we have considered two pairs of discriminative variables: the first one constituted of the LR and the goodput (cf. Section 4.1) and the second constituted of the LR and a TCP-compatible bandwidth share measure (cf. Section 4.2). Both LR and bandwidth characteristics (goodput or TCP-compatible) are clearly relevant not only as network characteristics but also as video quality parameters.

5.2.2. Similarity measure

Two kinds of measures should be defined: the similarity measure between two observed reports x and y ($d(x, y)$) and between an observed report x and a cluster C ($d(x, C)$). The former similarity measure can stand for the simple L^p distance ($d(x, y) = \sqrt[p]{\sum_i (x_i - y_i)^p}$) or any other more sophisticated distance suitable to a particular application. The retained similarity measure used in this work is given by $d(x, y) = \max_i (abs(x_i - y_i)/dt_i)$, where dt_i is a chosen threshold for the dimension i. The latter similarity measure is more difficult to apprehend. The simplest way is to choose in each cluster a representative $\hat{x}_C$ and to assign the distance $d(x, \hat{x}_C)$ to the distance between the point and the cluster ($d(x, C) = d(x, \hat{x}_C)$). We can also define the distance to cluster as the distance to the nearest or the furthest point of the cluster ($d(x, C) = \min_{y \in C} d(x, y)$ or

$d(x, C) = \max_{y \in C} d(x, y)$). The distance can also be a likelihood derived over a model mixture approach. The type of measure used will impact over the shape of the cluster and over the classification.

5.2.3. Classification algorithm

Each cluster is represented by a representative point and a weight. The representative point can be seen as a vector, the components of which are given by the discriminative variables considered in the clustering process.

The clustering algorithm is initialized with a maximal number of classes ($N_{\max}$) and a cluster creation threshold (d_{th}). AAs regularly receive RTCP reports from receivers and/or other AAs in their coverage area as described in Section 5.1. To classify the RRs in the different clusters, we use a very simple nearest neighbor (NN) k-means clustering algorithm (see pseudocode shown in Algorithm 1). Even if this algorithm might be subject to largely reported deficiencies as false clustering, dependencies on the order of presentation of samples, and nonoptimality which has lead researchers to develop more complex clustering mechanism as mixture modelling, we believe that this rather simple algorithm attain the goal of our approach which is to filter out RRs to a compact classification in a distributed, asynchronous way. A new report joins the cluster that has the lowest Euclidean (L^2) distance to it and updates the cluster representative by a weighted average of the points in the cluster. When a new point joins a cluster, it changes slightly the representative point which is defined as the cluster center and updates the weight of the cluster; afterwards, the point is dropped to achieve compression. If this minimal distance is more than a predefined threshold, a new cluster is created. This bounds the size of the cluster. We also use a maximal number of clusters (or classes) which is fixed to 5, as it is not realistic to have more layers in such a layered multicast scheme.

At the end of each reporting round, the resulting classification is sent back to the higher level AA (i.e., the manager) in the form of a vector of clusters representatives and of their associated weights, and clusters are reset to a null weight. Clusters received by different lower level AAs are classified following a similar clustering algorithm which will aggregate representative points of clusters, that is, cluster center, with the given weight. This amounts to applying the NN clustering algorithm to the representative points reported in the new coming RR.

At the higher level of the aggregators hierarchy, the clustering generated by aggregating lower level aggregator reports is renewed at the beginning of each reporting round.

As explained before, the classification of receivers should also integrate the recent history of receivers. This memory is introduced into the clustering process by using the cluster obtained during the past reporting round as an a priori in the highest level of the aggregator hierarchy.

Nevertheless, since, over large time scales, the stationary hypothesis cannot be always validated, a procedure must be added to ensure that we forget about far past-time reports and not to bias the cluster representative by out-of-date reports. This is handled by an exponential weighting heuristic: at each reporting round, the weight of a cluster is reduced by a constant factor (see Algorithm 2). If the weight of a cluster falls below a cluster suppression threshold level, the cluster is removed.

Search for the nearest cluster $d(\mathbf{r}, \hat{C}) = \min_C d(\mathbf{r}, C)$
if $(d(\mathbf{r}, \hat{C}) \geq d_{th})$
  if (Number of existing cluster $< N_{\max}$)
    Add a new cluster C_{new} and set $\hat{C} = C_{\text{new}}$
Recalculate the representative of cluster $\hat{C}$,

$$\hat{x}_{\hat{C}} = \frac{\text{weight}(\hat{C})\hat{x}_{\hat{C}} + \mathbf{r}}{\text{weight}(\hat{C}) + 1}$$

Increment the weight of cluster $\hat{C}$
d_{th} = predefined threshold
$N_{\max}$ = maximal number of clusters (5)
$\mathbf{r}$ = received receiver report

Algorithm 1: NN clustering algorithm.

At the beginning of each reporting round
  for all clusters C
    % *Weight the current normalized cluster by* γ
    weight(C) = weight$(C) * \gamma$
    if weight$(C) < w_{\min}$
      Remove cluster C
  Aggregate new normalized reports
  Send aggregate reports to the sender
$w_{\min}$ = predefined cluster suppression threshold
γ = memory weight

Algorithm 2: Aggregation algorithm at the highest level with memory weighting.

5.2.4. Cluster management

The clustering algorithm implements three mechanisms to manage the number of clusters: a cluster addition, a cluster removal, and a cluster merge mechanisms. The cluster addition and the cluster removal mechanisms have been described before. The cluster merging mechanism aims at reducing the number of clusters by combining two clusters that have been driven very close to each other. The idea behind this mechanism is that clusters should fill up uniformly the space of possible reception behaviors. The cluster merging mechanism merges two clusters that have a distance lower than a quarter of the cluster creation threshold (d_{th}). The distance between the two clusters is defined as the weighted distance of the cluster representatives. The merging threshold is chosen based on the heuristic that (1) d_{th} defines the fair diameter of a cluster and (2) two clusters that are distant by $d_{\text{th}}/4$ may be created by merging a cluster of diameter smaller than d_{th}. The cluster merging mechanism replaces the two clusters with a new cluster

represented by a weighted average of the two cluster representatives and a weight corresponding to the sum of the two clusters.

The combination of these three mechanisms of cluster management creates a very dynamic and reactive representation of the reception behaviour observed during the multicast session.

6. LAYERED SOURCE CODING RATE CONTROL

The feedback channel created by the clustering mechanism offers periodically to the sender information about the network state. More precisely, this mechanism delivers a LR, a bandwidth limit, and the number of receivers within a given cluster. This information is in turn exploited to optimize the number of source layers, the coding mode, the rate, and the level of protection of each layer. This section first describes the media and FEC rate control algorithm that takes into account both the network state and the source rate-distortion characteristics. The FGS video source encoding system used and the structure of the streaming server considered are then described.

6.1. Media and FEC rate-distortion optimization

We consider, in addition, the usage of FEC. In the context of transmission on the Internet, error detection is generally provided by the lower layer protocols. Therefore, the upper layers have to deal mainly with erasures or missing packets. The exact position of missing data being known, a good correction capacity can be obtained by systematic maximal distances separable (MDS) codes [32]. An (n, k) MDS code takes k data packets and produces $n - k$ redundant data packets. The MDS property allows to recover up to $n - k$ losses in a group of n packets. The effective loss probability $P_{\text{eff}}(k)$ of an MDS code, after channel decoding, is given by

$$P_{\text{eff}}(k) = P_e\left(\sum_{j=0}^{k-1}\binom{n-1}{j}P_e^{n-1-j}(1-P_e)^j\right), \tag{5}$$

where P_e is the average loss probability on the channel. One question to be solved is then, given the effective loss probability, how to split in an optimal way the available bandwidth for each layer between raw and redundant data. This amounts to finding the level of protection (or the code parameter k/n) for each layer.

The rates for both raw data and FEC (or equivalently, the parameter k/n) are optimized jointly as follows. For a maximum number of layers L supported by the source, the number of layers, their rate, and their level of protection are chosen in order to maximize the overall PSNR seen by all the receivers. Note that the rates are chosen in the set of N requested rates (feedback information). This can be expressed as

$$(\Omega_1, \ldots, \Omega_l) = \arg\max_{(\Omega_1,\ldots,\Omega_l)} G, \tag{6}$$

where $\Omega_i = (r_i, \kappa_i/n)$, $i = 1, \ldots, l$, with r_i representing the cumulated source and channel rate and κ_i/n the level of protection for each layer i. The quality measure G to be maximized is defined as

$$G = \sum_{j=1}^{N}\left(\sum_{i=1}^{l}\text{PSNR}(\Omega_i)\cdot P_{j,i}\right)\cdot C_j, \tag{7}$$

where

$$l = \arg\max_{k\in[1,\ldots,L]}\left\{\sum_{i=1}^{k} r_i \le R_j\right\}. \tag{8}$$

The terms R_j and C_j represent, respectively, the requested rate and the number of receivers in the cluster j. The term $\text{PSNR}(\Omega_i)$ denotes the PSNR increase associated with the reception of the layer i. Note that the PSNR corresponding to a given layer i depends on the lower layers. The term $P_{j,i}$ denotes the probability, for receivers of cluster j, that the i layers are correctly decoded and can be expressed as

$$P_{j,i} = \prod_{k=1}^{i}\left(1 - \tilde{p}_{\text{eff}_{j,k}}\left(\frac{\kappa_k}{n}\right)\right), \tag{9}$$

where $\tilde{p}_{\text{eff}_{j,k}}$ is the effective loss probability observed by all the receivers of the cluster j receiving the k considered layers. The values $\text{PSNR}(\Omega_i)$ are obtained by estimating the rate-distortion $D(R)$ performances of the source encoder on a training set of sequences. The model can then be refined on a given sequence during the encoding process, if the coding is performed in real time, or stored on the server in the case of streaming applications.

The upper complexity bound, in the case of an exhaustive search, is given by $L!/N!(N - L)!$, where L is the maximum number of layers and N the number of clusters. However, this complexity can be significantly reduced by first sorting the rates R_j requested by the different clusters. Once the rates R_j have been sorted, the constraint given by (8) allows to limit the search space of the possible combinations of rate r_i per layer. Hence, the complexity of an exhaustive search within the resulting set of possible values remains tractable. For large values of L and N, the complexity can be further reduced by using dynamic programming algorithm [33].

Notice that here we have not considered the use of hierarchical FEC. The FEC used here (i.e., MDS codes) are applied on each layered separately. Only their rates k_i/n are optimized jointly. The algorithm could be extended by using layered FEC as described in [34].

6.2. Fine-grain scalable source

The layers are generated by an MPEG-4 FGS video encoder [8, 9]. FGS has been introduced in order to cope with the adaptation of source rates to varying network bandwidths in the case of streaming applications with pre-encoded streams.

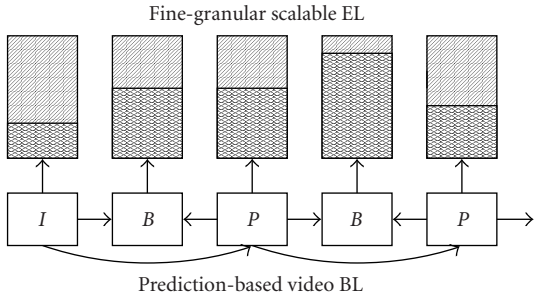


Figure 2: FGS video coding scalable structure.

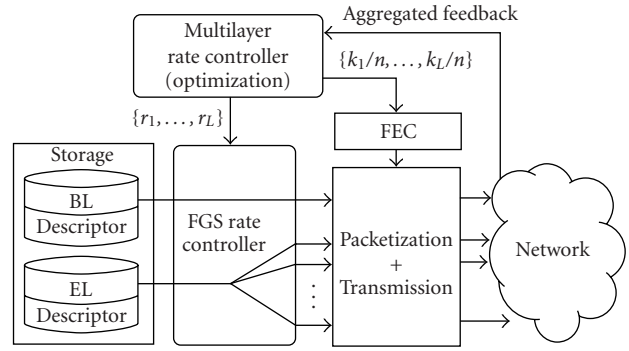


Figure 3: Multicast FGS streaming server.

Indeed, even if classical scalable (i.e., SNR, spatial, and temporal) coding schemes provide elements of response to the problem of rate adaptation to network bandwidth, those approaches suffer from limitations in terms of adaptation granularity. The structure of the FGS method is depicted in Figure 2. The BL is encoded at a rate denoted by R_{BL}, using a hybrid approach based on a motion compensated temporal prediction followed by a DCT-based compression scheme. The EL is encoded in a progressive manner up to a maximum bit rate denoted by R_{EL}. The resulting bitstream is progressive and can be truncated at any points, at the time of transmission, in order to meet varying bandwidth requirements. The truncation is governed by the rate-distortion optimization described above, considering the rate-distortion characteristics of the source. The encoder compresses the content using any desired range of bandwidths $[R_{\min} = R_{\mathrm{BL}}, R_{\max}]$. Therefore, the same compressed streams can be used for both unicast and multicast applications.

6.3. Multicast FGS streaming server

The experiments reported in this paper are done assuming an FGS streaming server. Figure 3 shows the internal structure of the multicast streaming system considered including the layered rate controller and the FEC module. For each video sequence prestored on the server, we have two separate bitstreams (i.e., one for BL and one for EL) coupled with its respective descriptors. These descriptors contain various information about the structure of the streams. Hence, it contains the offset (in bytes) of the beginning of each frame within the bitstream of a given layer. The descriptor of the BL contains also the offset of the beginning of a slice (or video packet) of an image. The composition timestamp (CTS) of each frame used as the presentation time at the decoder side is also contained in the descriptor.

Upon receiving a new list $(r_0, r_1, \ldots, r_L)$ of rate constraints, the FGS rate controller computes a new bit budget per frame (for each expected layer) taking into account the frame rate of the video source. Then, at the time of transmission, the FGS rate controller partitions the FGS enhancement into a corresponding number of "sublayers." Each layer is then sent to a different IP multicast group. Notice that, regardless of the number of FGS ELs that the client subscribes to, the decoder has to decode only one EL (i.e., the sublayers of the EL merge at the decoder side).

6.4. Rate control signalling

In addition to the value of the RTT computed for the probe RTT packets, the RTCP SRs periodically sent include information about the sent layers, that is, their number, their rate, and their level of protection, according to the following syntax:

(i) NL: an 8-bit field which gives the number of enhancement layers;
(ii) BL: a 16-bit field which gives the rate of the base layer;
(iii) EL_i: a set of 16-bit fields which give the rate of the enhancement layer i, $i \in 1, \ldots, NL$;
(iv) k_i: a set of 8-bit fields conveying the rate of the Reed-Solomon code used for the protection of layer i, $i \in 0, \ldots, NL$.[1]

7. EXPERIMENTAL RESULTS

The performance of the SARC algorithm has been evaluated considering various sets of discriminative clustering variables using the NS2 (version 2.1b6), network simulator.

7.1. Analysis of fairness

The first set of experiments aimed at analyzing the fairness of the flows produced against conformant TCP flows. Fairness has been analyzed using the single bottleneck topology shown in Figure 4. In this topology, a number of sending nodes are connected to many receiving nodes via a common link with a bottleneck rate of 8 Mbps and a delay of 50 milliseconds. The video flows controlled by the SARC protocol are competing with 15 conformant TCP flows. Figure 5a depicts the respective throughput of one video

[1] Here we consider Reed-Solomon codes of rates k/n. The value of n is fixed at the beginning of the session and only the parameter k is adapted dynamically during the session. However, we could also easily consider adapting the parameter n, therefore the syntax of the SR packet would have to be extended accordingly.

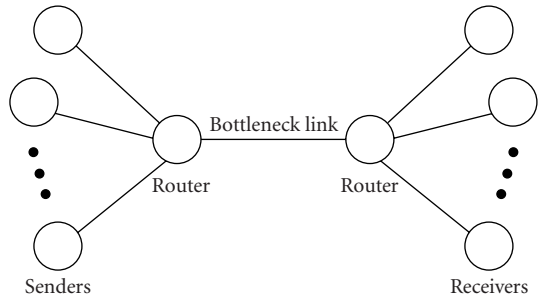


Figure 4: Simulation topology (bottleneck).

flow controlled with the goodput measure and of two out of the 15 TCP flows. Figure 5b depicts the throughputs obtained when using the TCP-compatible rate equation. As expected, the flow regulated with the goodput measure does not compete fairly with the TCP flows (cf. Figure 5a). In the presence of cross traffic at high rate, the EB decreases regularly to reach the lower bound $R_{\min}$ that has been set to 256 Kbps. The average throughput of the flow regulated with the TCP-compatible measure matches closely the average TCP throughput with a smoother rate (cf. Figure 5b).

7.2. Loss rate and PSNR performances

The second set of experiments aimed at measuring the PSNR and LR performances of the rate control mechanism, with two measures (goodput and TCP-compatible measures), with and without the presence of FEC. We have considered the multicast topology shown in Figure 6. The periodicity of the feedback rounds is set to be equal to the maximum RTT value of the set of receivers. The sequence used in the experiments, called "Brest,"[2] has a duration of 300 seconds (25 Hz, 6700 frames). The rate-distortion characteristics of the FGS source is depicted in Figure 7. The experiments depicted here are realized with the MoMuSys MPEG-4 version 2 video codec [9].

7.2.1. Testing scenario

Given the topology of the multicast tree, we have considered a source representation on three layers, each layer being transmitted to an IP multicast address. The BL is encoded at a constant bit rate of 256 Kbps. The overall rate (base layer plus two ELs) ranges from 256 Kbps up to 1 Mbps. At $t = 0$, each client subscribes to the three layers with respective initial rates of $R_{\mathrm{BL}} = 256$ Kbps, $R_{\mathrm{EL1}} = 100$ Kbps, and $R_{\mathrm{EL2}} = 0$ Kbps. During the session, the video stream has to compete with point-to-point UDP cross traffic with a constant bit rate of 192 Kbps and with TCP flow. These competing flows contribute to a decrease of the links bottleneck. The activation of the cross traffic between clients represented by "squares" on Figure 6, in the time interval from 100 to 200 seconds, limits the bottleneck of the corresponding link (i.e., LAN 1's client) down to 320 Kbps. Similarly, competing TCP traffic is generated between clients denoted by "triangles" in the interval from 140 to 240 seconds leading to a bottleneck rate of the link (i.e., LAN 4's clients) down to 192 Kbps during the corresponding time interval.

The first test aimed at showing the benefits for the quality perceived by the receivers of an overall measure that would also take into account the source characteristics (and in particular the rate-distortion characteristics) versus a simple optimization of the overall goodput. Thus, we compare our results with the SAMM algorithm proposed in [3]. The corresponding mechanism is called SAMM-like in the sequel.

The SARC algorithm, relying on the rate-distortion optimization, has then been tested with, respectively, the goodput and the TCP-compatible measures in order to evidence the benefits of the TCP-compatible rate control in this layered multicast transmission system. In the sequel, these approaches are, respectively, called goodput-based source adaptive rate control (GB-SARC) and TCP-friendly source adaptive rate control (TCPF-SARC). The constant K is set to 4 in the experiments. In addition, in order to evaluate the impact of the FEC, we have considered the TCP-compatible bandwidth estimation both with and without FEC (TCPF-SARC+FEC) for protecting the BL. When FEC is not applied, the k_i parameter of each layer is set to n (i.e., 10 in the experiments).

7.2.2. Results

Figures 8 and 9 show the results obtained with the SAMM-like algorithm. It can be seen that the SAMM-like approach does not permit an efficient usage of the bandwidth. For example, the LAN 2's client (with a link with a bottleneck rate of 768 Kbps) has not received more than 300 Kbps on its link. Similar observations can be done with receivers of other LANs. Notice also that if the rate had not been lower bounded by an $R_{\min}$ value, the goodput of the different receivers would have converged to a very small value. In addition to the highly suboptimal usage of bandwidth, the approach suffers from a very unstable behavior in terms of subscriptions and unsubscriptions to multicast groups.

Figures 10, 11, and 12 show the rate variations of the different layers of the FGS source over the session, obtained, respectively, with the GB-SARC, TCPF-SARC, and TCPF-SARC+FEC methods. Figures 13, 14, and 15 depict the throughput estimated with these three methods versus the real measures of goodput, the LR, the number of layers received, and the PSNR values observed for two representative clients (i.e., LAN 2 with a bottleneck rate of 768 Kbps and LAN 4 with a bottleneck rate of 384 Kbps).

Figures 10 and 13, with the GB-SARC algorithm, show that the rate control that takes into account the PSNR (or rate-distortion) characteristics of the source leads to a better bandwidth utilization than the SAMM-like approach. In addition, the throughput estimated follows closely the bottleneck rates of the different links. Moreover, the number of irrelevant subscriptions and unsubscriptions to multicast

[2] Courtesy of Thomson Multimedia R&D France.

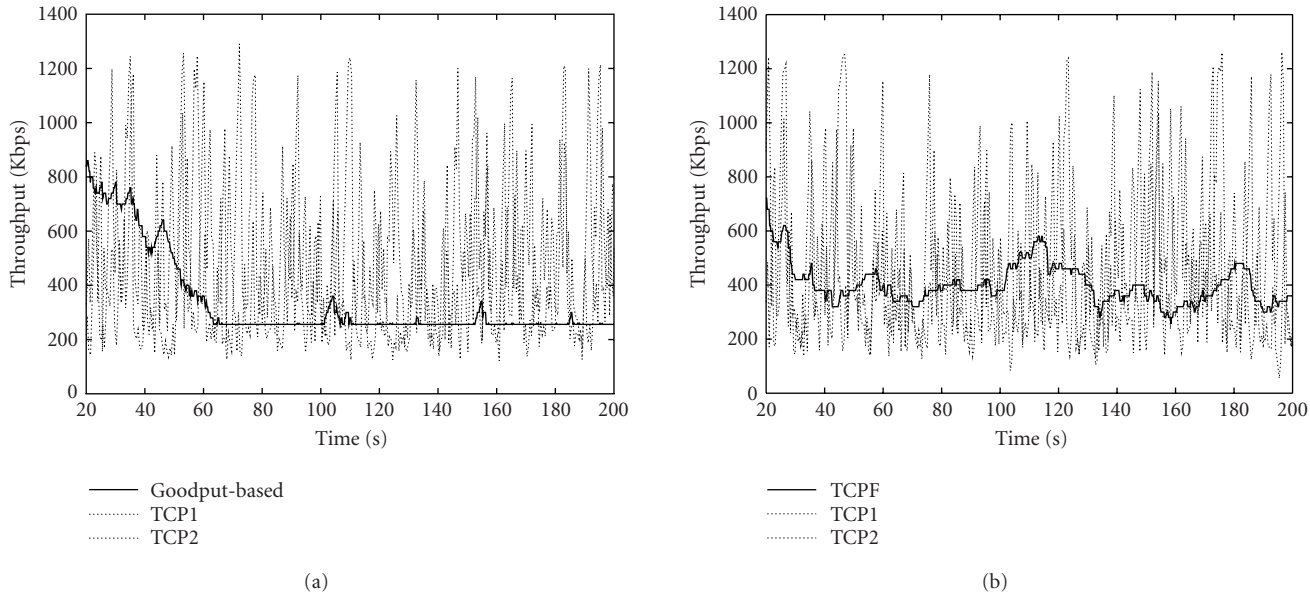


Figure 5: Respective throughputs of two TCP flows and of one rate-controlled flow with (a) a measure of goodput and (b) the TCP-compatible measure.

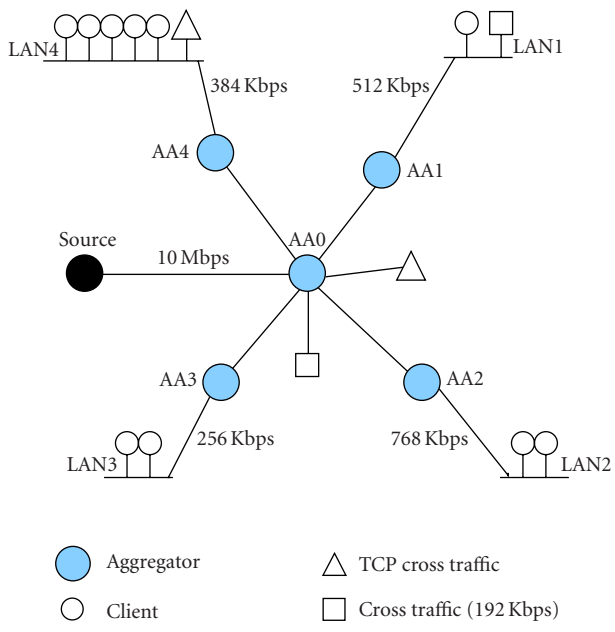


Figure 6: Simulated topology.

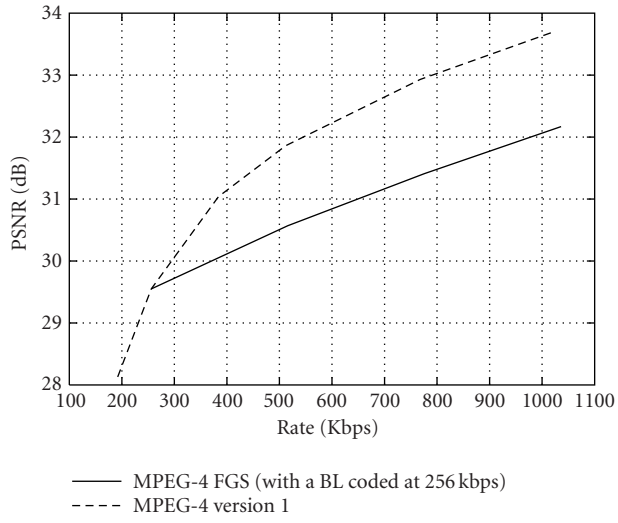


Figure 7: Rate-distortion model of the FGS video source.

groups is strongly reduced. However, the LRs observed remain high. For example, the LAN 4's client observe an average LR of 30% between 240 seconds and 300 seconds. This is due to the fact that during this time interval, the receiver of LAN 1 (bottleneck rate of 512 Kbps) has subscribed to the first enhancement layer (EL1), hence the rate of this layer is higher than the bottleneck rate of the LAN 4's clients. In this case, the GB-SARC algorithm does not permit a reliable bandwidth estimation for the LAN 4's clients. As expected, the quality of the received video suffers from the high LRs and the obtained PSNR values are relatively low. Finally, another important drawback is that during the corresponding period, the rate constraints given to the FGS video streaming server are very unstable (see Figure 10).

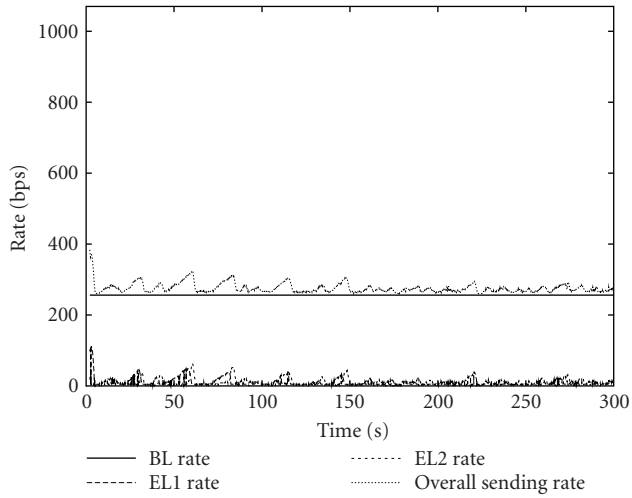
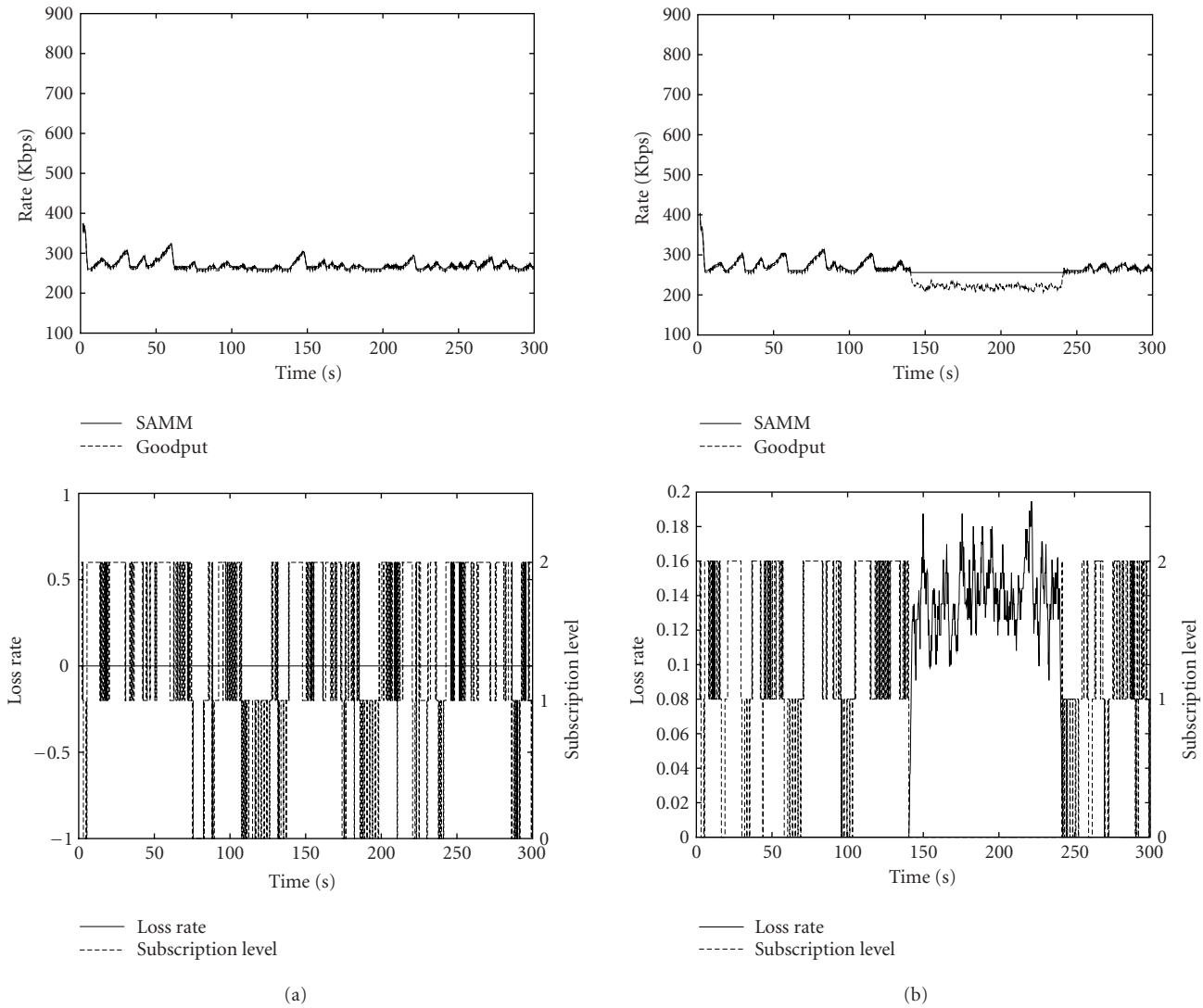


Figure 8: Rate variations for each layer of the FGS video source with the SAMM-like approach.



(a) (b)

Figure 9: SAMM-like throughput versus real goodput measure, LR, and subscription level obtained for (a) a LAN 2's client (link 768 Kbps) and (b) a LAN 4's client (link 384 Kbps).

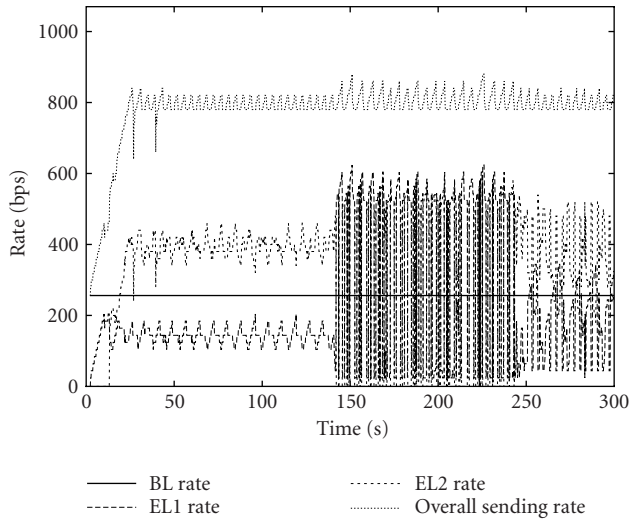


Figure 10: Rate variations for each layer of the FGS video source with the GB-SARC approach.

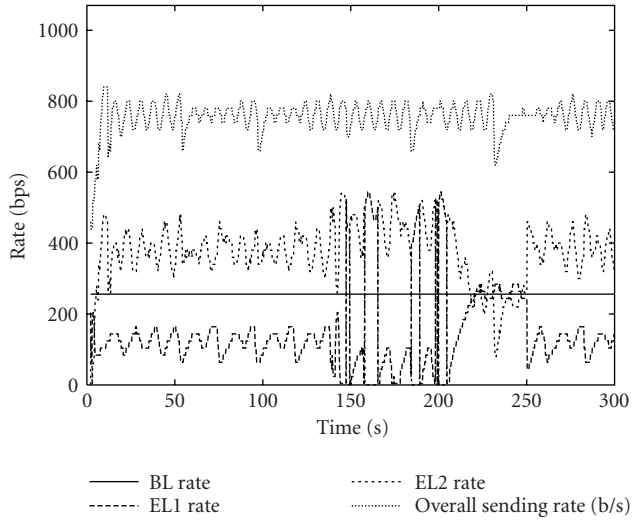


Figure 11: Rate variations for each layer of the FGS video source with the TCPF-SARC approach.

With the TCPF-SARC algorithm (cf. Figures 11 and 14), the sending rates of the different layers follows closely the variations of the bottleneck rates of the different links. This leads to stable sessions with low LRs and with a restricted number of irrelevant subscriptions and unsubscriptions to multicast groups. The comparison of the PSNR curves in Figure 14 reveals a gain of at least db for LAN 2 with respect to LAN 4. This evidences the interest of such multilayered rate control algorithm in a multicast heterogeneous environment. Notice that the peaks of instantaneous LRs observed result from a TCP-compatible prediction which occasionally exceeds the bottleneck rate. Also, in Figure 14b, the LR observed over the time interval from 140 to 240 seconds remains constant and relatively high. This comes from the fact that, in the presence of competing traffic, the bottleneck rate available for the video source is lower than the rate of the BL which in the particular case of an FGS source is maintained constant in average (e.g., 256 Kbps).

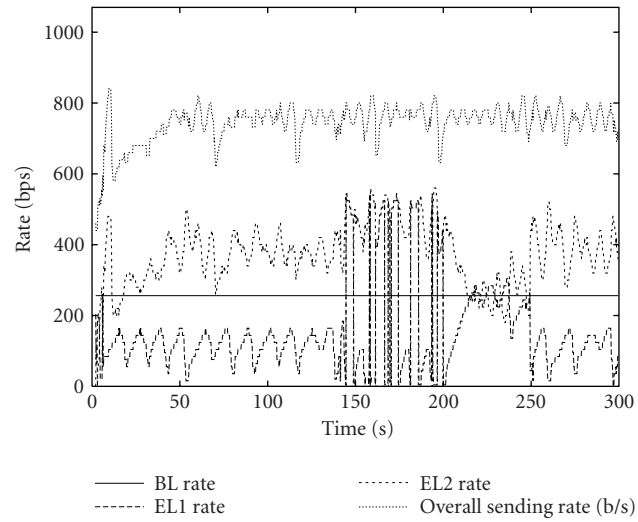


Figure 12: Rate variations for each layer of the FGS video source with the TCPF-SARC + FEC approach.

The FEC permits improving slightly the PSNR performances, especially for the receivers of LAN4 (cf. Figure 15b). It can be seen on Figure 12 that the usage of FEC however leads to a bit more unstable behavior, that is, to higher rate fluctuations of the different layers of the FGS source.

8. CONCLUSION

In this paper, we have presented a new multicast multilayered congestion control protocol called SARC. This algorithm relies on an FGS layered video transmission system in which the number of layers, their rate, as well as their level of protection are adapted dynamically in order to optimize the end-to-end QoS of a multimedia multicast session. A distributed clustering mechanism is used to classify receivers according to the packet LR and the bandwidth estimated on the path leading to them. Experimentation results show the ability of the mechanism to track fluctuation of the available bandwidth in the multicast tree, and at the same time the capacity to handle fluctuating LRs. We have shown also that using LR and TCP-compatible measures as discriminative variables in the clustering mechanism leads to higher overall PSNR (hence QoS) performances than using the LR and goodput measures.

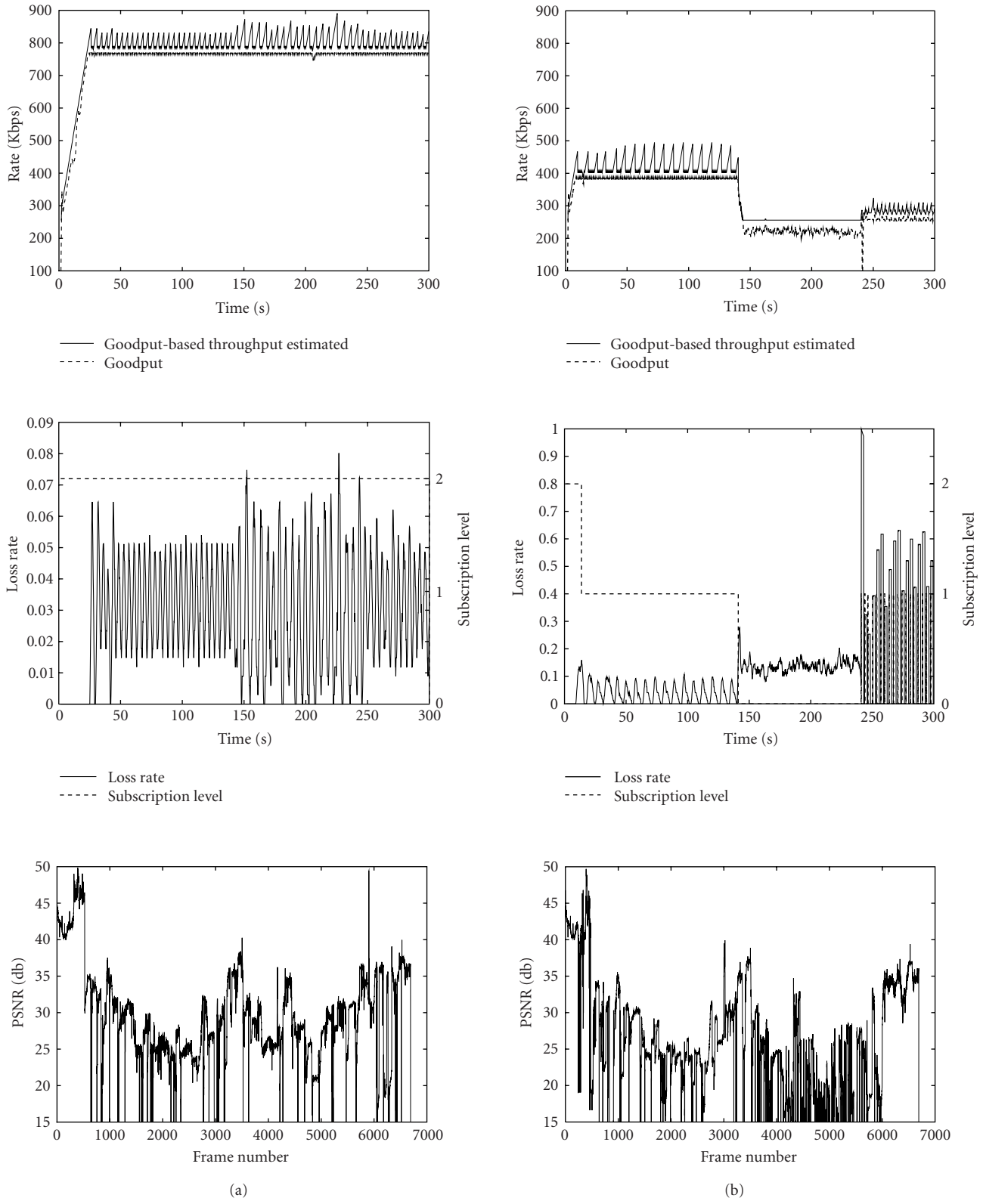


Figure 13: GB-SARC throughput versus real goodput measure, LR, subscription level, and PSNR obtained for (a) a LAN 2's client (link 768 Kbps) and (b) a LAN 4's client (link 384 Kbps).

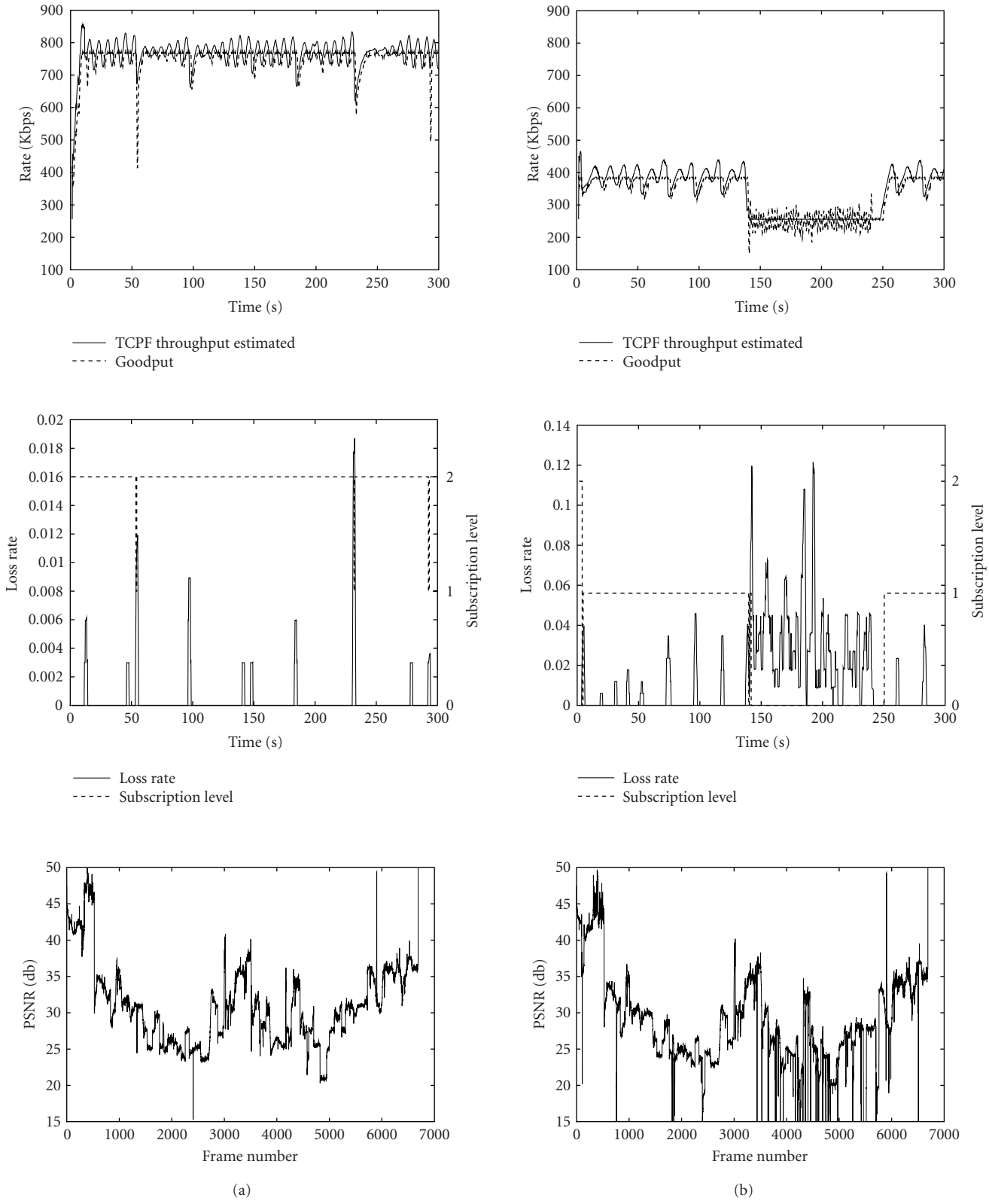


Figure 14: TCPF-SARC throughput versus real goodput measure, LR, subscription level, and PSNR obtained for (a) a LAN 2's client (link 768 Kbps) and (b) a LAN 4's client (link 384 Kbps).

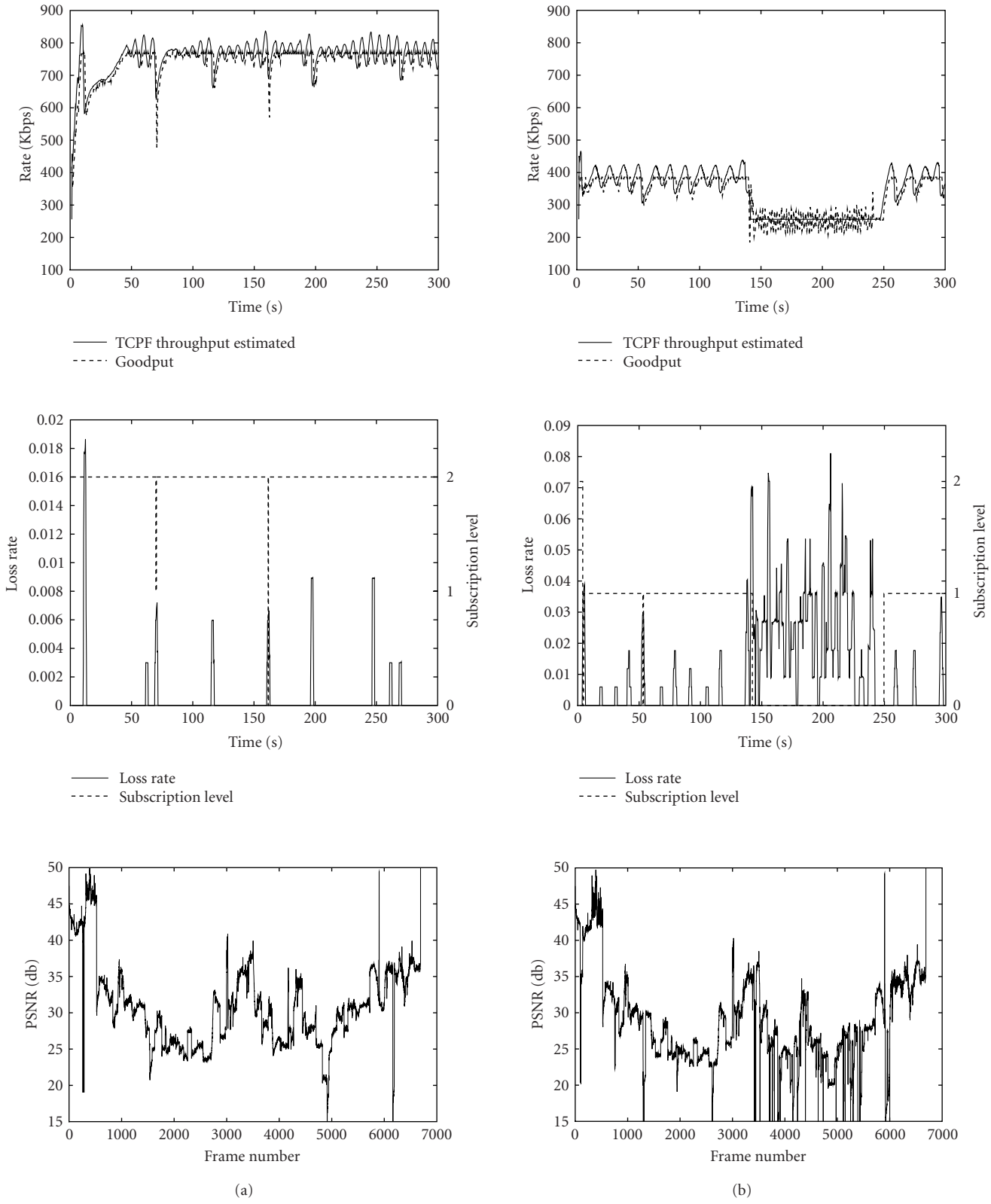


Figure 15: TCPF-SARC throughput with FEC versus real goodput measure, LR, subscription level, and PSNR obtained for (a) a LAN 2's client (link 768 Kbps) and (b) a LAN 4's client (link 384 Kbps).

REFERENCES

[1] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. Conference of the Special Interest Group on Data Communication (ACM SIGCOMM '96)*, pp. 117–130, Stanford, Calif, USA, August 1996.

[2] T. Turletti, S. Fosse-Parisis, and J. C. Bolot, "Experiments with a layered transmission scheme over the internet," Tech. Rep. RR-3296, INRIA, Sophia-Antipolis, 1997.

[3] B. J. Vickers, C. Albuquerque, and T. Suda, "Source adaptive multi-layered multicast algorithms for real-time video distribution," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 720–733, 2000.

[4] D. Sisalem and A. Wolisz, "MLDA: A TCP-friendly congestion control framework for heterogeneous multicast environments," Tech. Rep., GMD FOKUS, Berlin, Germany, 2000.

[5] Y. Wang and Q. F. Zhu, "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, 1998.

[6] J. C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for internet telephony," in *Proc. Conference on Computer Communications (IEEE Infocom '99)*, pp. 1453–1460, NY, USA, March 1999.

[7] K. Salamatian, "Joint source-channel coding applied to multimedia transmission over lossy packet network," in *Proc. Packet Video Workshop (PV '99)*, NY, USA, April 1999.

[8] H. Radha and Y. Chen, "Fine granular scalable video for packet networks," in *Proc. Packet Video Workshop (PV '99)*, Columbia University, NY, USA, April 1999.

[9] Mobile Multimedia Systems (MoMuSys) Software, "MPEG-4 video verification model 4.1", December 2000.

[10] J. C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *Proc. Conference of the Special Interest Group on Data Communication (ACM SIGCOMM '94)*, pp. 58–67, London, UK, September 1994.

[11] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 982–1001, 1997.

[12] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," in *Proc. Conference on Computer Communications (IEEE Infocom '98)*, pp. 996–1003, San Francisco, Calif, USA, March 1998.

[13] D. Sisalem and A. Wolisz, "MLDA: A TCP-friendly congestion control framework for heterogeneous multicast environments," in *Proc. International Workshop on Quality of Service (IWQoS '00)*, Pittsburgh, Pa, USA, June 2000.

[14] X. Hénocq, F. Le Léannec, and C. Guillemot, "Joint source and channel rate control in multicast layered video transmission," in *Proc. SPIE International Conference on Visual Communication and Image Processing (VCIP '00)*, pp. 296–307, Perth, Australia, June 2000.

[15] A. Legout and E. W. Biersack, "Pathological behaviors for RLM and RLC," in *Proceedings of International Conference on Network and Operating System Support for Digital Audio and Video (NOSSDAV '00)*, pp. 164–172, Chapel Hill, NC, USA, June 2000.

[16] A. Legout and E. W. Biersack, "PLM: Fast convergence for cumulative layered multicast transmission schemes," in *Proc. ACM (SIGMETRICS '00)*, pp. 13–22, Santa Clara, Calif, USA, 2000.

[17] S. Bhattacharya, D. Towsley, and J. Kurose, "The loss path multiplicity problem in multicast congestion control," in *Proc. Conference on Computer Communications (IEEE Infocom '99)*, vol. 2, pp. 856–863, NY, USA, March 1999.

[18] J. Widmer and M. Handley, "Extending equation-based congestion control to multicast applications," in *Proc. Conference of the Special Interest Group on Data Communication (ACM SIGCOMM '01)*, pp. 275–286, San Diego, Calif, USA, August 2001.

[19] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. Conference of the Special Interest Group on Data Communication (ACM SIGCOMM '00)*, pp. 43–56, Stockholm, Sweden, August 2000.

[20] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shave, "FLID-DL: Congestion control for layered multicast," in *Proc. Second International Workshop on Networked Group Communication (NGC '00)*, pp. 71–81, Palo Alto, Calif, USA, November 2000.

[21] M. Luby and V. k. Goyal, "Wave and equation based rate control building block," Internet Engineering Task Force, Internet Draft draft-ietf-rmt-bb-webrc-04, June 2002.

[22] Q. Guo, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "A sender-adaptive and receiver-driven layered multicast scheme for video over internet," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '01)*, Sydney, Australia, May 2001.

[23] K. Salamatian and T. Turletti, "Classification of receivers in large multicast groups using distributed clustering," in *Proc. Packet Video Workshop (PV '01)*, Taejon, Korea, May 2001.

[24] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP thoughput: a simple model and its empirical validation," in *Proc. Conference of the Special Interest Group on Data Communication (ACM SIGCOMM '98)*, pp. 303–314, University of British Columbia, Vancouver, Canada, August 1998.

[25] J. Viéron and C. Guillemot, "Real-time constrained TCP-compatible rate control for video over the internet," to appear in IEEE Transactions on Multimedia.

[26] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the MBone multicast network," in *Proc. IEEE Global Internet Conference*, London, UK, November 1996.

[27] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves, "Organizing multicast receivers deterministically by packet-loss correlation," in *Proc. 6th ACM International Conference on Multimedia (ACM Multimedia 98)*, Bristol, UK, September 1998.

[28] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal On Selected Areas in Communications*, vol. 15, no. 3, pp. 407–421, 1997.

[29] R. El-Marakby and D. Hutchison, "Scalability improvement of the real-time control protocol (RTCP) leading to management facilities in the internet," in *Proc. 3rd IEEE Symposium on Computers and Communications (ISCC '98)*, pp. 125–129, Athens, Greece, June 1998.

[30] K. L. Calvert, J. Griffioen, B. Mullins, A. Sehgal, and S. Wen, "Concast: Design and implementation of an active network service," *IEEE Journal on Selected Area in Communications (JSAC)*, vol. 19, no. 3, pp. 720–733, 2001.

[31] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantiser design," *IEEE Transactions on Communications*, vol. 28, pp. 84–95, January 1980.

[32] F. J. Mac Williams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, North Holland, Amsterdam, 1977.

[33] D. Koo, *Elements of Optimization*, Springer-Verlag, NY, USA, 1977.

[34] D. Tan and A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 373–387, 2001.

Jérôme Viéron received his M.S. degree in computer science from the University of Rennes, France, in 1999. From 1999 to 2003, he was pursuing his Ph.D. works at INRIA. He received his Ph.D. degree in computer science from the University of Rennes, France, in 2003. Currently he is with the Corporate Research Center of Thomson Multimedia R&D in Rennes, France. He works in the Multimedia Streaming & Storage Lab. His research interests are new generation scalable video compression for TV, HDTV, and digital cinema.

Thierry Turletti received his M.S. and Ph.D. degrees in computer science, both from the University of Nice Sophia-Antipolis, France, in 1990 and 1995, respectively. He has done his Ph.D. studies in the RODEO group at INRIA Sophia Antipolis. During 1995–1996, he was a Postdoctoral Fellow in the Telemedia, Networks and Systems Group at the MIT Laboratory for Computer Science (LCS), Massachusetts Institute of Technology (MIT). He is currently a Research Scientist at the Planète group at INRIA Sophia Antipolis. His research interests include multimedia applications, congestion control, and wireless networking. Dr. Turletti currently serves on the editorial board of Wireless Communications and Mobile Computing.

Kavé Salamatian is an Associate Professor at Paris VI University in France and conducts his researches at LIP6. His main areas of research are networking information theory and Internet measurement and modelling. He is actually the coordinator of a large research effort in Internet measurement and modelling in France. He has graduated in 1998 from Paris-SUD Orsay University with a Ph.D. degree in computer science. He worked during his Ph.D. on joint source-channel coding applied to multimedia transmission over Internet. Dr. Salamatian also has an M.S. in theoretical computer science from Paris XI University (1996) and an M.S. in communication engineering from Isfahan University of Technology (1995).

Christine Guillemot is currently "Directeur de Recherche" at INRIA, in charge of a research group dealing with image modelling, processing, and video communication. She holds a Ph.D. degree from Ecole Nationale Supérieure des Telecommunications (ENST), Paris. From 1985 to October 1997, she has been with CNET France Telecom where she has been involved in various projects in the domain of coding for TV, HDTV, and multimedia applications. From January 1990 to mid 1991, she has worked at Bellcore, NJ, USA, as a Visiting Scientist. Her research interests are signal and image processing, video coding, and joint source and channel coding for video transmission over the Internet and over wireless networks. She currently serves as Associated Editor for IEEE Transactions on Image Processing.

EURASIP Journal on Applied Signal Processing 2004:2, 176–191

Fine-Grained Rate Shaping for Video Streaming over Wireless Networks

Trista Pei-chun Chen
NVIDIA Corporation, Santa Clara, CA 95050, USA
Email: tchen@nvidia.com

Tsuhan Chen
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA
Email: tsuhan@cmu.edu

Received 30 November 2002; Revised 14 October 2003

Video streaming over wireless networks faces challenges of time-varying packet loss rate and fluctuating bandwidth. In this paper, we focus on streaming precoded video that is both source and channel coded. Dynamic rate shaping has been proposed to "shape" the precompressed video to adapt to the fluctuating bandwidth. In our earlier work, rate shaping was extended to shape the channel coded precompressed video, and to take into account the time-varying packet loss rate as well as the fluctuating bandwidth of the wireless networks. However, prior work on rate shaping can only adjust the rate coarsely. In this paper, we propose "fine-grained rate shaping (FGRS)" to allow for bandwidth adaptation over a wide range of bandwidth and packet loss rate in fine granularities. The video is precoded with fine granularity scalability (FGS) followed by channel coding. Utilizing the fine granularity property of FGS and channel coding, FGRS selectively drops part of the precoded video and still yields decodable bitstream at the decoder. Moreover, FGRS optimizes video streaming rather than achieves heuristic objectives as conventional methods. A two-stage rate-distortion (RD) optimization algorithm is proposed for FGRS. Promising results of FGRS are shown.

Keywords and phrases: fine-grained rate shaping, rate shaping, fine granularity scalability, rate-distortion optimization, video streaming.

1. INTRODUCTION

Due to the rapid growth of wireless communication, video over wireless network has gained a lot of attention [1, 2, 3]. However, wireless network is hostile for video streaming because of its time-varying error rate and fluctuating bandwidth. Wireless communication often suffers from multipath fading, intersymbol interference, and additive white Gaussian noise, and so forth; thus, the error rate varies over time. In addition, the bandwidth of the wireless network is also time varying. Therefore, it is important for a video streaming system to address these issues.

Joint source-channel coding (JSCC) techniques [4, 5] are often applied to achieve error-resilient video transport with *online coding*. Given the bandwidth requirement, the joint source-channel coder seeks the best allocation of bits for the source and channel coders by varying the coding parameters. However, JSCC techniques are not suitable for streaming *precoded* video. The precoded video is both source and channel coded prior to transmission. The network conditions are not known at the time of coding. "Rate shaping," which was called dynamic rate shaping (DRS) in [6, 7, 8], was proposed to solve the bandwidth adaptation problem. DRS "shapes," that is, reduces the bit rate of the single-layered pre source coded (pre-compressed) video to meet the real-time bandwidth requirement. DRS adapts the bandwidth by dropping either high-frequency coefficients of each block or by dropping several blocks in a frame.

To protect the video from transmission errors, source coded video bitstream is often protected by forward error correction (FEC) codes [9]. Redundant information, known as parity bits, is added to the original source coded bits, assuming that systematic codes are adopted. Conventional DRS did not consider shaping for the parity bits in addition to the source coding bits. In our earlier work, we extended rate shaping for streaming the precoded video that is both pre-source-and-channel coded [10]. Such a scheme was called "baseline rate shaping (BRS)." BRS can be applied to precoded video that is source coded with H.263 [11], MPEG-2 [12], or MPEG-4 [13] scalable coding and channel coded with Reed-Solomon codes [9] or rate-compatible punctured convolutional (RCPC) codes [14]. By means of

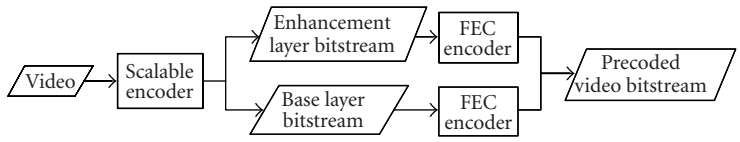


Figure 1: System diagram of the precoding process: scalable encoding followed by FEC encoding.

discrete rate-distortion (RD) combination, BRS chooses the best state, which corresponds to a certain way to drop part of the precoded video, to satisfy the bandwidth constraint.

The state chosen by BRS, however, only allows for coarse bandwidth adaptation capability. In this paper, we adopt MPEG-4 fine granularity scalability (FGS) [15] for source coding, and erasure codes [9, 16] for FEC coding. Unlike conventional scalability modes such as signal-to-noise ratio (SNR) scalability, MPEG-4 FGS generates a bitstream that is partially decodable over a wide range of bit rates. The more bits the FGS decoder receives, the better the decoded video quality is. On the other hand, it has been known that erasure codes are still decodable if the number of erasures is within the error/loss protection capability of the codes. Therefore, the proposed "fine-grained rate shaping (FGRS)," which is based on the fine granularity property of FGS and erasure codes, allows for fine rate shaping. Moreover, the proposed FGRS optimizes video streaming rather than achieves heuristic objectives such as unequal packet loss protection (UPP). A two-stage (RD) optimization algorithm is proposed. Note that FGRS focuses on the transport aspect as opposed to the coding aspect of video streaming.

The two-stage RD optimization is designed to find the solution fast and optimally. In Stage 1, a model-based hypersurface is trained with a small set of rate and distortion pairs to approximate the relationship between all rate and distortion pairs. The solution of Stage 1 can be found in the intersection in which the hypersurface meets the bandwidth constraint. In Stage 2, the near-optimal solution from Stage 1 is refined with the hill-climbing-based approach. We can see that Stage 1 aims to find the optimal solution globally with the model-based hypersurface and Stage 2 refines the solution locally.

This paper is organized as follows. In Section 2, we introduce BRS for bandwidth adaptation of the precoded video, which is both scalable and FEC coded. Discrete RD combination algorithm is applied to deliver the best video quality. In Section 3, FGRS is proposed for streaming the FEC coded FGS bitstream. We first formulate the RD optimization problem then provide a two-stage RD optimization algorithm to solve the problem. In Section 4, experiments are carried out to show the superior performance of the proposed FGRS. Concluding remarks are given in Section 5.

2. BASELINE RATE SHAPING

We propose to use BRS to reduce the bit rate of the precoded video, which is both source and channel coded, given the time-varying error rate and bandwidth. Unlike JSCC techniques that allocate the bits for the source and channel coders by varying the coding parameters, BRS performs bandwidth adaptation for the *precoded* video at the time of delivery. BRS decision, as to select which part of the precoded video to drop, varies from time to time. There is no need to reencode as JSCC with different source and channel coder parameters at later time with a different channel condition. Only a different BRS decision needs to be made for the same bitstream. In addition, rate shaping can be applied to adapt to the network condition of each link along the path of transmission from the sender to the receiver. This is in particular suitable for wireless video streaming since wireless networks are heterogeneous in nature. One single joint source-channel coded bitstream cannot meet the needs of all the links along the path of transmission. Rate shaping can optimize video streaming for each link.

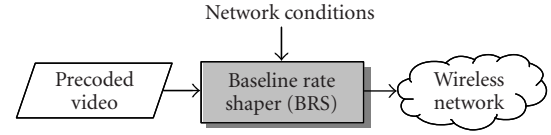


Figure 2: Streaming of the precoded video with BRS.

We start by giving the system description of BRS then provide the algorithm for RD optimization.

2.1. System description of video streaming with baseline rate shaping

Video streaming consists of three stages from the sender to the receiver: (i) precoding, (ii) streaming with rate shaping, and (iii) decoding, as shown in the following from Figure 1 to Figure 3.

The precoding process (Figure 1) refers to source coding using scalable video coding [11, 12, 13] and FEC coding. Scalable video coding yields prioritized video bitstream. The concept of rate shaping works for any prioritized video bitstream in general.[1] Without loss of generality, we consider SNR scalability. Reed-Solomon codes [9] are used as the FEC codes in this paper.

[1] For example, in DRS [6], bits that carry the information of the low-frequency DCT coefficients are ranked with high priorities in the video bitstream, as opposed to the ones that carry the information of the high-frequency DCT coefficients. By means of data partitioning, the single-layered nonscalable coded bitstream can have different priorities among different segments of the video bitstream.

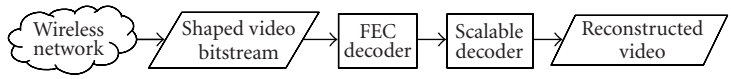


Figure 3: System diagram of the decoding process: FEC decoding followed by scalable decoding.

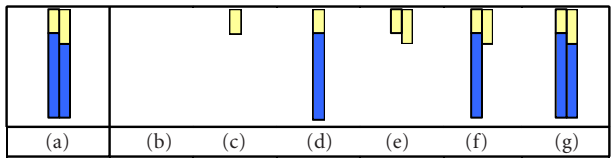


Figure 4: (a) All four segments of the precoded video and (b)–(g) valid states of BRS: (b) state $(0,0)$, (c) state $(1,0)$, (d) state $(1,1)$, (e) state $(2,0)$, (f) state $(2,1)$, and (g) state $(2,2)$.

In Figure 2, the pre-source-and-channel coded bitstream is then passed through BRS to adjust its bit rate before being sent to the wireless network. BRS will perform bandwidth adaptation considering the given packet loss rate in an RD optimized manner. The distortion here is described by the mean square error (MSE) of the decoded video. Packet loss rate, instead of bit error rate (BER), is considered since the shaped precoded video will be transmitted in packets.

The decoding process (Figure 3) consists of FEC decoding followed by scalable decoding. The task of rate shaping is performed in the sender and/or midway gateways/routers.

2.2. Discrete rate-distortion optimization algorithm

BRS reduces the bit rate of each *decision unit* of the precoded video before it sends the precoded video to the wireless network. A decision unit can be a frame, a macroblock, and so forth, depending on the granularity of the decision. We use a frame as the decision unit herein. BRS performs two kinds of RD optimizations with (i) mode decision and (ii) discrete RD combination, depending on how much delay the rate shaping decisions can allow. We will discuss both mode decision and discrete RD combination in the following.

(a) *BRS by mode decision*

We consider the case in which the video is scalable coded into two layers: one base layer and one enhancement layer. These two layers are FEC coded with UPP. That is, the base layer is FEC coded with stronger packet loss protection. Therefore, there are four *segments* in the precoded video. The first segment consists of the bits of the base layer video bitstream (upper-left segment of Figure 4a). The second segment consists of the bits of the enhancement layer video bitstream (upper-right segment of Figure 4a). The third segment consists of the parity bits for the base layer video bitstream (lower-left segment of Figure 4a). The fourth segment consists of the parity bits for the enhancement layer video bitstream (lower-right segment of Figure 4a). BRS decides a subset of the four segments to send. Note that some constraints need to be imposed for a valid subset. For example, if the segment that consists of the parity bits for the base layer video bitstream is selected, the segment that consists of the bits of the base layer video bitstream must be selected as well. In the case of two layers of video bitstream, six valid combinations are shown in Figures 4b, 4c, 4d, 4e, 4f, and 4g. We call each valid combination a *state*. Each state is represented by a pair of integers (x, y), where x is the number of segments selected counting from the segment consisting of the bits of the base layer, and y is the number of segments selected counting from the segment consisting of the parity bits for the base layer. Note that x counts from the base layer because the enhancement layer cannot be decoded without the base layer; y counts from the base layer because the base layer needs to be protected by parity bits more than the enhancement layer. The two integers x and y satisfy the relationship of $x \geq y$.

Each state has its RD performance represented by a dot in the RD map, such as the ones shown in Figures 5a and 5b. The state constellations are different for different frames because of variations in video content and packet loss rate for different frames. If the bandwidth requirement is "B" for each frame, BRS performs mode decision by selecting the state that has the least distortion. For example in Figure 5, state $(1,1)$ of Frame 1 and state $(2,0)$ of Frame 2 are chosen.

(b) *BRS by discrete RD combination*

By allowing some delay in making the rate shaping decision, BRS can optimize video streaming with a better overall quality. By allowing some delay, we mean to accumulate the total bandwidth for a group of pictures (GOP) and to allocate the bandwidth intelligently among frames in a GOP. Video is typically coded with variable bit rate in order to maintain a constant video quality. We want to allocate different numbers of bits for different frames in a GOP to utilize the total bandwidth more efficiently.

Assume that there are F frames in a GOP and the total bandwidth budget for these F frames is C. Let $x(i)$ be the state (represented by a pair of integers mentioned in (a)) chosen for frame i, and let $D_{i,x(i)}$ and $R_{i,x(i)}$ be the resulting distortion and rate allocated at frame i, respectively. The goal of the rate shaper is to minimize

$$\sum_{i=1}^{F} D_{i,x(i)} \tag{1}$$

subject to

$$\sum_{i=1}^{F} R_{i,x(i)} \leq C. \tag{2}$$

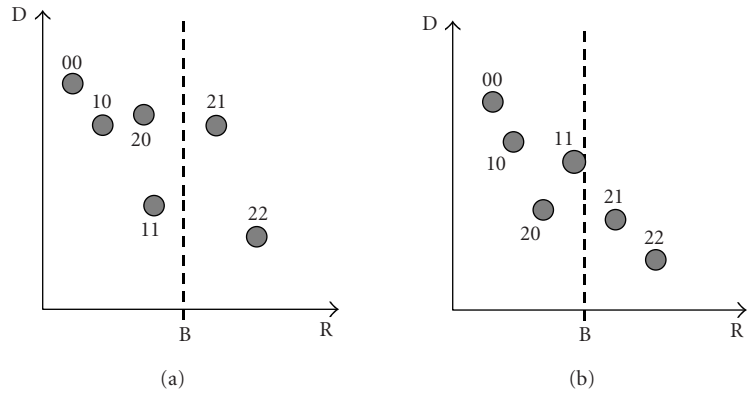


Figure 5: RD maps of (a) Frame 1, (b) Frame 2.

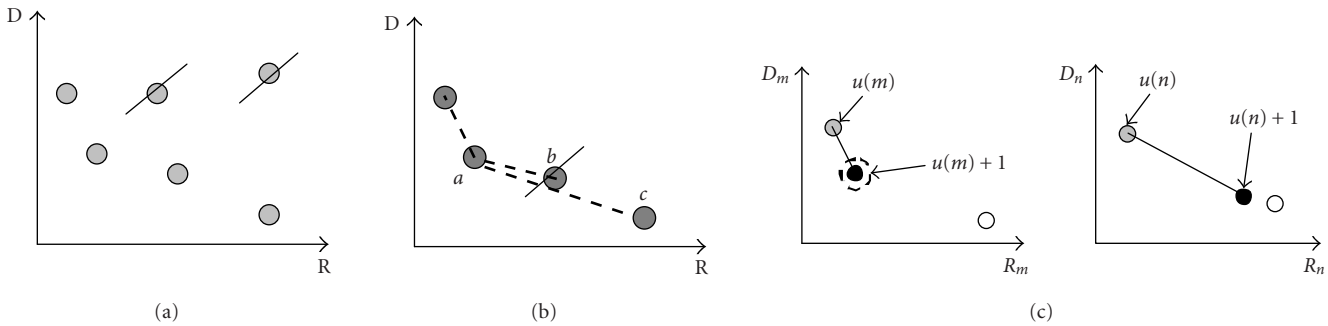


Figure 6: Discrete RD combination algorithm: (a) and (b) elimination of states inside the convex hull of each frame, and (c) allocation of rate to the frame m that utilizes the rate more efficiently.

The discrete RD combination algorithm [10, 17] finds the solution by first eliminating the states that are inside the convex hull (Figures 6a and 6b) for each frame. The algorithm then allocates the rate step by step to the frame that utilizes the rate more efficiently. That is, among frame m and frame n, if frame m gives a better ratio than frame n regarding distortion decrease over rate increase by moving from the current state $u(m)$ to the next state $u(m)+1$, then the rate is allocated to frame m (the next state $u(m)+1$ of frame m is circled in Figure 6c) from the available total bandwidth budget. The allocation process continues until the total bandwidth budget has been consumed completely.

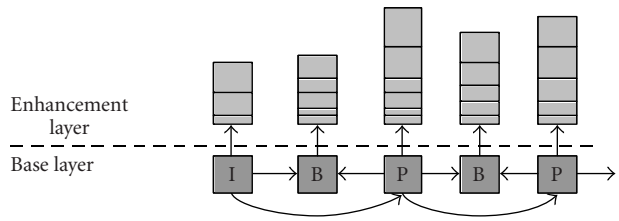


Figure 7: Dependency graph of the base layer and FGS enhancement layer. Base layer has temporal prediction with P and B frames. Enhancement layer is encoded with reference to the base layer only.

3. FINE-GRAINED RATE SHAPING (FGRS)

As mentioned, BRS performs the bandwidth adaptation for the precoded video by selecting the best state of each frame at any given packet loss rate. Since the packet loss rate and the bandwidth at any given time could lie in any value over a wide range of values, we want to extend the notion of rate shaping to allow for finer grained decisions. There then prompts the need for source and channel coding techniques that offer fine granularities in terms of video quality and packet loss protection, respectively.

FGS has been proposed to provide bitstreams that are still decodable when truncated at any byte interval. That is, FGS enhancement layer bitstream is decodable at any rate provided with an intact base layer bitstream. With such a property, FGS was adopted by MPEG-4 for streaming applications [15]. Figure 7 illustrates two layers of video bitstream: the base layer and the FGS enhancement layer. The base layer is predictive coded while the FGS enhancement layer only uses the corresponding base layer as the reference.

On the other hand, it has been known that the erasure codes provide "fine-grained" packet loss protection with

more and more symbols[2] received at the FEC decoder [9, 16]. The "shaped" erasure code is still decodable if the number of erasures/losses from the transmission is no more than $d_{\min} - 1$ (number of unsent symbols), where $d_{\min}$ is the minimum distance of the code. An erasure code can successfully decode the message with the number of erasures up to $d_{\min} - 1$, considering both the unsent symbols and the losses taken place in the transmission. Therefore, the more symbols are sent, the better the sent bitstream can cope with the losses. In this paper, we use Reed-Solomon codes as the erasure codes as mentioned in Section 2. In Reed-Solomon codes, $d_{\min} - 1$ equals $n - k$, where k is the message size in symbols and n is the code size in symbols. Thus, the partial code with size $r \leq n$ is still decodable if the number of losses from the transmission is no more than $r - k$.

3.1. System description of video streaming with fine-grained rate shaping

Similar to BRS, there are three stages for transmitting the video from the sender to the receiver: (i) precoding, (ii) streaming with rate shaping, and (iii) decoding, as shown in Figures 8, 9, and 10.

Through MPEG-4 encoding, two layers of bitstream are generated: one base layer and one FGS enhancement layer (Figure 7). We will consider hereafter the bandwidth adaptation and packet loss resilience for the FGS enhancement layer bitstream only, assuming that the base layer bitstream is reliably transmitted as shown in Figure 9b or is considered by approaches outside the scope of this paper. The general rule is to perform enhancement layer bandwidth adaptation after the base layer is reliably transmitted. The enhancement layer bitstream will not enhance the quality of the video if its reference base layer is corrupted. Otherwise, a drift prevention remedy is needed.

Recalling that we use a frame as the decision unit, we look at the FGS enhancement layer bitstream of a frame. FGS enhancement layer bitstream consists of bits of all the bit planes of this frame. The most significant bit plane (MSB plane) is coded before the less significant bit planes until the least significant bit plane (LSB plane). In addition, since the data in each bit plane is variable-length coded (VLC), if some part of a bit plane is corrupted (due to packet losses), the remaining part of the bit plane becomes undecodable. Bits at the beginning of the enhancement layer bitstream of a frame is more important than the following bits.

Before appending the parity symbols to the FGS enhancement layer bitstream, we first divide all the symbols (in this paper, each symbol consists of 14 bits) for this frame into several *sublayers* (Figure 11a). The way to divide the symbols into sublayers is arbitrary except that the later sublayers are longer in length than the previous ones, that is $k_1 \geq k_2 \geq \cdots \geq k_h$, since we want to achieve UPP. A natural way to construct the sublayers is to let Sublayer 1 consist of symbols of the MSB plane, Sublayer 2 consist of symbols of the MSB-1 plane, ..., and Sublayer h consist of symbols of the LSB plane. Each sublayer is then FEC encoded with erasure codes to the same length n (Figure 11b). The lower portions of the stripes in Figure 11b consist of the parity symbols. The precoded video is stored and can be used later at the time of delivery.

At the transport stage, FEC coded FGS bitstream is passed through FGRS for bandwidth adaptation, given the current packet loss rate. Note that FGRS is different from JSCC-like approaches, which perform FEC encoding for the FGS bitstream at the time of delivery with a bit allocation scheme that achieves certain objectives, as proposed by Radha and van der Schaar [18, 19, 20] and Yang et al. [21]. That is, FGRS focuses on the transport aspect as opposed to the coding aspect. Moreover, FGRS optimizes video streaming rather than achieves certain objectives. We will elaborate on the optimization algorithm taken later.

3.2. Fine-grained rate shaping

With the precoded video, bandwidth adaptation can be implemented naively by dropping the symbols in the order shown in Figure 12a. Given a certain bandwidth requirement for this frame, Sublayer 1 has more parity symbols kept than Sublayer 2 and so on. Shaped bitstream with such a bandwidth adaptation scheme has UPP to the sublayers. We will refer to this method as "UPPRS" herein. However, such UPPRS scheme might not be optimal. We propose FGRS (Figure 12b) for bandwidth adaptation given the current packet loss rate. The darken bars in Figure 12b are selected to be sent by FGRS.

We start from the problem formulation. A FGS enhancement layer bitstream provides better and better video quality as more and more sublayers are correctly decoded. In other words, the total distortion is decreased as more sublayers are correctly decoded. With Sublayer 1 correctly decoded, we reduce the total distortion by G_1 (accumulated *gain* is G_1); with Sublayer 2 correctly decoded, we reduce the total distortion further by G_2 (accumulated gain is $G_1 + G_2$), and so on. If Sublayer i is corrupted, the following Sublayers $i + 1$, $i + 2$, and so forth, become undecodable. Note that gain G_i of Sublayer i can either (i) be calculated, given the FGS bitstream, after performing partial decoding; or (ii) be embedded in the bitstream as the "metadata." Gain G_i of Sublayer i is different for every frame.

Since the precoded video is transmitted over error prone wireless networks, sublayers are subject to loss and have certain recovery rates given a particular rate shaping decision. The *expected* accumulated gain is then

$$G = \sum_{i=1}^{h} \left(G_i \prod_{j=1}^{i} v_j \right), \tag{3}$$

where h is the number of sublayers of this frame and v_j is the recovery rate of Sublayer j, which is a function of r_j as will be shown later. Sublayer j is recoverable (or successfully decodable) if the number of erasures resulting from the lossy

[2] "Symbols" are used instead of "bits" since the FEC codes use a symbol as the encoding/decoding unit. In this paper, we use 14 bits for one symbol. The selection of the symbol size in bits depends on the user.

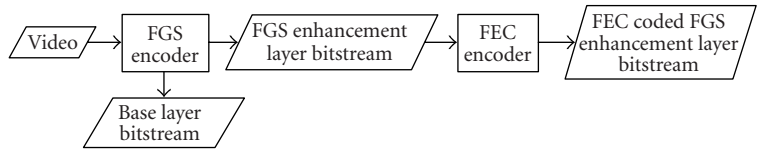


Figure 8: System diagram of the precoding process: FGS encoding followed by FEC encoding.

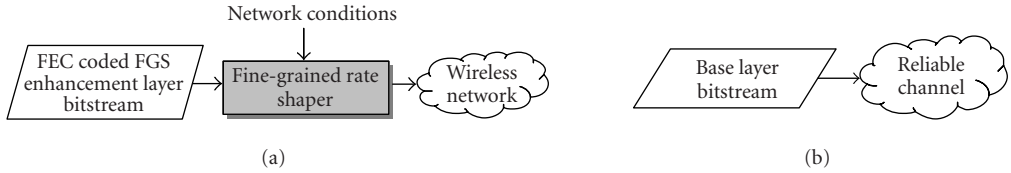


Figure 9: Transport of the precoded bitstreams: (a) transport of the FEC coded FGS enhancement layer bitstream with rate shaper via the wireless network and (b) transport of the base layer bitstream via the reliable channel.

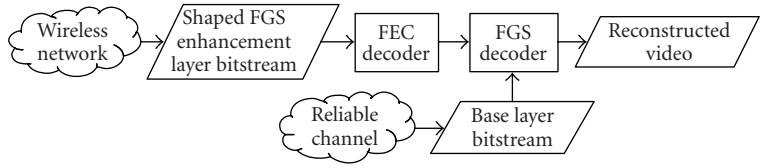


Figure 10: System diagram of the decoding process: FEC decoding followed by FGS decoding.

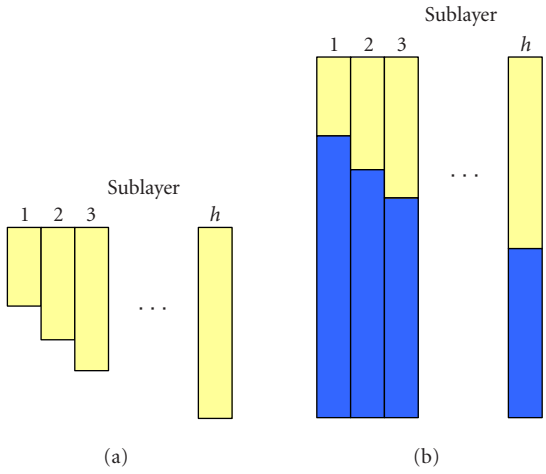


Figure 11: Precoded video: (a) FGS enhancement layer bitstream in sublayers and (b) FEC coded FGS enhancement layer bitstream.

transmission is no more than $r_j - k_j$; k_j is the message (the symbols from the FGS bitstream) size of Sublayer j, and r_j is the number of symbols selected to be sent for Sublayer j. The recovery rate v_j is the summation of the probabilities that no loss occur, one erasure occurs, and so on until $r_j - k_j$ erasures occur:

$$v_j = \sum_{l=0}^{r_j - k_j} p\{l\}, \quad j = 1 \sim h, \tag{4}$$

where l is the number of erasures that occur. If each erasure occurs as a Bernoulli trial with probability e_m, the probability of having l erasures out of r_j symbols is

$$p\{l\} = \binom{r_j}{l} (e_m)^l (1 - e_m)^{r_j - l}. \tag{5}$$

The symbol loss rate can be derived from the packet loss rate as $e_m = 1 - (1 - e_p)^{m/s}$, where s is the packet size and m is the symbol size in bits. Depending on the error model (Bernoulli trial, two-state Markov model, etc.), (5) can be replaced with different probability functions.

By choosing different combinations of the number of symbols for each sublayer, the expected accumulated gain will be different. The rate-shaping problem can then be formulated as follows: maximize

$$G = \sum_{i=1}^{h} \left(G_i \prod_{j=1}^{i} v_j \right) \tag{6}$$

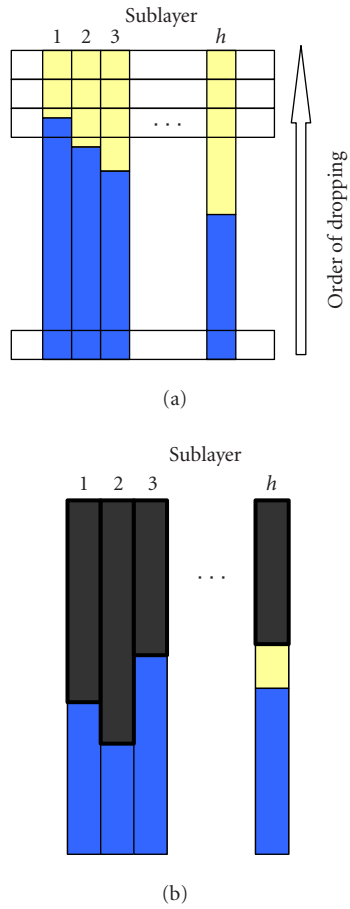


Figure 12: Bandwidth adaptation with (a) UPPRS and (b) FGRS. The part represented by darken bars are selected to be sent by FGRS.

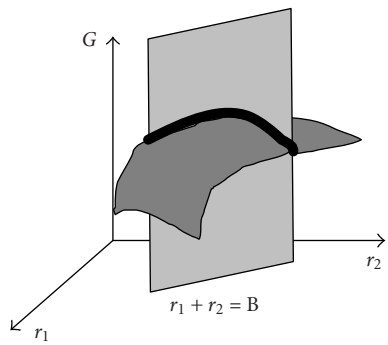


Figure 13: Intersection of the model-based hypersurface (dark surface) and the bandwidth constraint (gray plane), illustrated with $h = 2$.

subject to

$$\sum_{i=1}^{h} r_i \le B. \tag{7}$$

To solve the problem, an exhausted search on all possible combinations of $\mathbf{r} = [r_1 \ r_2 \ \cdots \ r_h]$ or hill-climbing-based approaches as described in [22, 23, 24], where RD optimization is made for automatic repeat request (ARQ) decisions, can be performed. We propose in this paper a two-stage RD optimization algorithm. The two-stage RD optimization algorithm first finds the *near-optimal* solution fast. The near-optimal solution is then *refined* by the hill climbing approach. The proposed two-stage RD optimization is different from [22, 23, 24] in three folds. First, the model-based Stage 1 allows us to examine fewer samples from all operational RD states. Only a small set of samples are needed to train the model needed for RD optimization. Second, the proposed distortion measure (or "expected accumulated gain" in the terminology of the paper) accounts for the effects of packet loss as well as the channel codes by means of recovery rates. Finally, the proposed two-stage RD optimization algorithm can avoid the problem that the solution could be trapped in the local maximum or reach the local maximum too slow. Due to the complexity consideration, Stage 2 can be skipped. Stage 1 does not just serve as a simple initialization stage. It can already find a near-optimal solution.

Packetization is performed after rate shaping. That is, symbols are grouped into packets after the decision of $\mathbf{r} = [r_1 \ r_2 \ \cdots \ r_h]$ has been made. Similar packetization method can be found in [20], while [25] applied bit errors on the bitstream directly. The packets can be sent with "user datagram protocol (UDP)" [26]. It is assumed that any error in the packet will result in a packet loss. More considerations on packetization can be found in UDP-Lite [27]. This paper focuses on rate shaping, assuming that the network condition is provided regardless of which specific packetization method is used.

(1) *Two-stage RD optimization: Stage 1*

We can see from (3) and (4) that the expected accumulated gain G is related to $\mathbf{r} = [r_1 \ r_2 \ \cdots \ r_h]$ implicitly through the recovery rates $\mathbf{v} = [v_1 \ v_2 \ \cdots \ v_h]$. We can instead find a model-based hypersurface that explicitly relates $\mathbf{r}$ and G. The model parameters can be trained from a set of training data $(\mathbf{r}, G)$, where $\mathbf{r}$ values are chosen by the user and G values can be computed from (3) and (4). The optimal solution is in the intersection (Figure 13) in which the model-based hypersurface meets the bandwidth constraint. A complex model, with a lot of parameters, can be used to describe as close as possible the true distribution of the RD states. The solution obtained with this model will be as close to optimal as possible. However, the number of $(\mathbf{r}, G)$ pairs needed to train the model-based hypersurface increases with the number of parameters.

In this paper, we use a quadratic equation to describe the relation between $\mathbf{r}$ and G as follows:

$$\hat{G} = \sum_{i=1}^{h} a_i r_i^2 + \sum_{i,j=1, i\neq j}^{h} b_{ij} r_i r_j + \sum_{i=1}^{h} c_i r_i + d. \tag{8}$$

To distinguish the hypersurface modeled $\hat{G}$ from the real expected gain G, we denote the former with a "head" sign. The model parameters a_i, b_{ij}, c_i, and d are trained differently for each frame. They can be solved by surface fitting with a set of training data $(\mathbf{r}, G)$ obtained by (3) and (4). For example, the parameters can be computed by

$$\begin{pmatrix} a_i\text{'s} \\ b_{ij}\text{'s} \\ c_i\text{'s} \\ d \end{pmatrix} = \left(R^T R\right)^{-1} R^T \begin{pmatrix} {}^1G \\ {}^2G \\ \vdots \\ {}^{\Xi}G \end{pmatrix}, \tag{9}$$

where the left super index of G is the index of the training data and R is a matrix consisting Ξ rows of $(r_i^2\text{'s}, r_i r_j\text{'s}, r_i\text{'s}, 1)$.

The complexity of computing a_i's, b_{ij}'s, c_i's, and d relates to the number of parameters $h^2 + h + 1$ and the number of training data Ξ, using (9). Note that the number of training data Ξ is in general much greater than the number of parameters $h^2 + h + 1$. Thus, a more complex model, such as a third-order model with $h^3 + h^2 + h + 1$ parameters, is not suitable since it requires much more training data than a quadratic model. In addition, second-order Taylor expansion can nicely approximate most functions. Equation (8) can be seen as a second-order approximation to (3). To reduce the computation complexity in reality, we can also choose a smaller h if the precoding process is also under our control (which is outside the scope of the rate shaper).

With (8), the near-optimal solution can be obtained by the use of Lagrange multiplier as follows:

$$\begin{aligned} J = &\left(\sum_{i=1}^{h} a_i r_i^2 + \sum_{i,j=1, i\neq j}^{h} b_{ij} r_i r_j + \sum_{i=1}^{h} c_i r_i + d \right) \\ &+ \lambda \left(\sum_{i=1}^{h} r_i - B \right). \end{aligned} \tag{10}$$

By $\partial J / \partial r_i = 0$, we get

$$r_i = \frac{-1}{2a_i} \left(\sum_{j=1, j\neq i}^{h} b_{ij} r_j + c_i + \lambda \right), \tag{11}$$

where

$$\lambda = \frac{2B + \sum_{i=1}^{h} (1/a_i) \left(\sum_{j=1, j\neq i}^{h} b_{ij} r_j + c_i \right)}{-\sum_{i=1}^{h} (1/a_i)}. \tag{12}$$

The near-optimal solution can be solved recursively using (11) and (12), starting from the initial condition that all sublayers are allocated with equal number of symbols, $r_1 = r_2 = \cdots = r_h = B/h$.

(2) *Two-stage RD optimization: Stage 2*

Stage 1 of the two-stage RD optimization gives a near-optimal solution. The solution can be refined by a hill-climbing-based approach (Algorithm 1). The solution from Stage 1 is perturbed in Stage 2 in order to yield a larger expected accumulated gain. The process can be iterated until the solution reaches a stopping criterion such as the convergence.

The idea of allocating bandwidth optimally for sublayers can be extended to a higher level to allocate bandwidth efficiently among frames in a GOP. The problem formulation is

```
While (stop == false)
    z^i = r^i  for all i = 1 ~ h
    For (j = 1; j <= h; j++)
        For (k = 1; k <= h; k++)
            z^k = z^k + delta for k == j //Increase sublayer j
            z^k = z^k - delta /(h - 1) for k! = j //Decrease others
        End
        Evaluate G_j
    End
    Find the j* with the largest G_j*.
    For (i = 1; i <= h; i++)
        r^i = r^i + delta for i == j*
        r^i = r^i - delta /(h - 1) for i! = j*
    End
    Calculate the stop criterion.
End
```

Algorithm 1: Pseudocodes of hill-climbing algorithm.

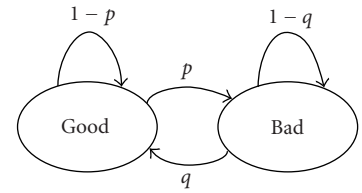


Figure 14: Two-state Markov chain for bit error simulation.

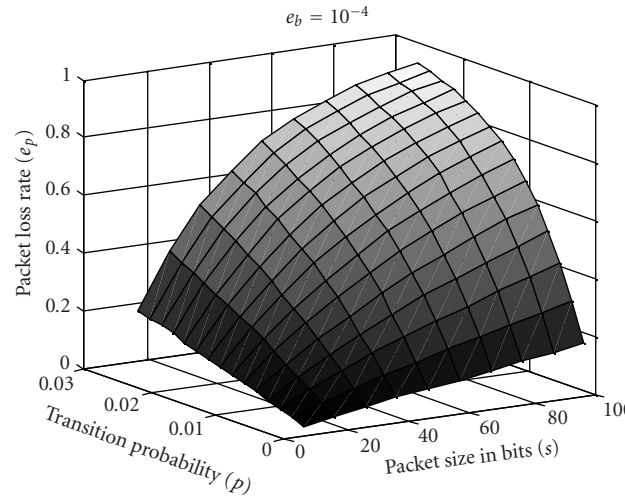


Figure 15: Packet loss rate as a function of the transition probability and the packet size.

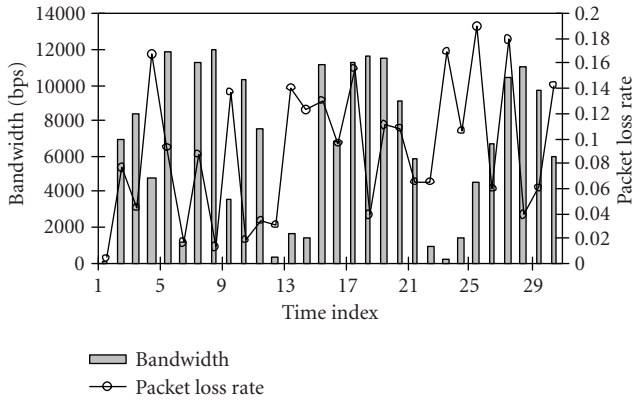


Figure 16: Network conditions: bandwidth and packet loss rate fluctuations.

slightly different from the original (6) as follows: maximize

$$G = \sum_{m=1}^{F} \left[\sum_{i=1}^{h} \left(G_{mi} \prod_{j=1}^{i} v_{mj} \right) \right] \tag{13}$$

subject to

$$\sum_{m=1}^{F} \sum_{i=1}^{h} r_{mi} \le C, \tag{14}$$

where F is the number of frames in a GOP. FGRS will incur delay with duration of F frames if it allows for optimization among frames in a GOP.

To summarize, the proposed FGRS achieves the best streaming performance for FEC coded FGS bitstream with the two-stage RD optimization. The two-stage RD optimization obtains the optimal solution by first finding the near-optimal solution, then refining the solution with a hill-climbing-based approach.

4. EXPERIMENT

We start by describing the wireless network simulation for the experiment. We then compare the proposed FGRS with the naive UPPRS described in Figure 12a.

4.1. Experiment setup

Wireless networks are generally associated with time-varying packet loss rate and fluctuating bandwidth. The packet loss rate and bandwidth vary at each time interval. We simulate random bandwidth fluctuation according to an autoregressive (AR) process [28] and use a two-state Markov model [29, 30] to simulate the bursty bit errors. The two-state Markov model is also adopted by [31, 32]. "Good" and "Bad" in Figure 14 correspond to error free and erroneous states of a bit, respectively. The BER e_b is related to the transition probabilities p and q by $e_b = p/(p+q)$.

Since the coded bitstream is transmitted in packets, let us look at how the packet loss rate e_p relates to the transition probability p and the BER e_b. With BER e_b, transition probability p, and packet size s, the packet loss rate of the s-bit packet is

$$e_p = 1 - (1 - e_b)(1 - p)^{s-1}. \tag{15}$$

We observe two properties from (15) given the same BER e_b: (i) the smaller the transition probability p, the smaller the packet loss rate e_p, and (ii) the smaller the packet size s, the smaller the packet loss rate e_p. These two properties are shown in Figure 15 with $e_b = 10^{-4}$.

Besides the two properties we have just seen, it is also known that to detect the loss of packets, some information such as the packet number has to be added to each packet. The smaller the packet is, the heavier the overhead is. Therefore, it is a trade-off between the selection of the packet size and the resulting packet loss rate. We use $s = 280$ (bits) in this paper. Users can select the packet size s according to real system consideration using (15).

The time-varying bandwidth is simulated pseudorandomly according to an AR process. The bandwidth available at current time t is fed to FGRS optimization of time $t+1$ in order to simulate the delay nature of the network feedback. Such delay in feedback will not affect too much the performance since the bandwidth requirements of the two consecutive frames are closely related, given the AR assumption. Example traces of simulated packet loss rate and bandwidth observed at the rate shaper are shown in Figure 16. The packet loss rate is plotted using the line and the bandwidth is illustrated using the vertical bars. Each interval in the axis of time index represents 0.33 seconds.

The test video sequences are "Akiyo," "Foreman," and "Stefan" in common intermediate format (CIF) (Figures 17a, 17b, and 17c). The frame rate is three frames/s.

Table 1: PSNR gains in Y, U, and V components with sequences Akiyo, Foreman, and Stefan.

PSNR gain (dB)	Y component	U component	V component
Akiyo	1.38	1.28	0.87
Foreman	0.86	0.44	0.52
Stefan	0.76	0.34	0.38

4.2. Experiment result

Results for sequence Akiyo are shown in Figures 18 and 19. Results for sequence Foreman is shown in Figures 20 and 21. Results for sequence Stefan is shown in Figures 22 and 23. The overall PSNR performance for all the three test sequences are listed in Figure 24 and Table 1. Results for different wireless channel conditions are shown in Figure 25.

Figures 18, 20, and 22 show how bit allocation with UPPRS and FGRS is done in bytes (converted from number of symbols) for each sublayer. After bit allocation, the number of symbols to send is constrained to be at least k_i for each sublayer (i.e., to satisfy $r_i \ge k_i$) by moving the number of symbols allocated for the higher sublayers to the lower layers that does not satisfy $r_i \ge k_i$ as shown in Algorithm 2.

(a) (b) (c)

Figure 17: Test video sequences in CIF: (a) Akiyo, (b) Foreman, and (c) Stefan.

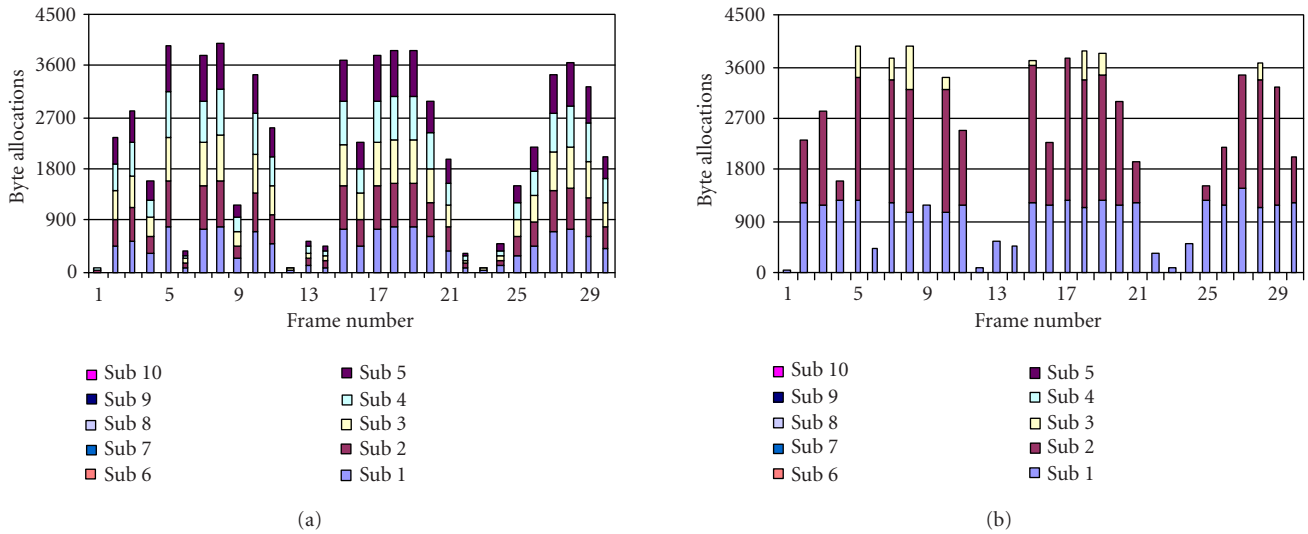


Figure 18: Sublayer byte allocations with sequence Akiyo by (a) UPPRS and (b) FGRS.

With limited bandwidth, FGRS allocates enough bytes to Sublayer 1 (indicated as sub 1 in the figures) first, than to Sublayer 2, and so on. Allocating enough bytes to a sublayer means providing enough packet loss protection, but not allocating too many bytes as to include too much redundancy. The bit allocation process happens automatically by the proposed two-stage RD optimization, considering the current packet loss rate and the bandwidth requirement.

From the frame-by-frame PSNR performance in Figures 19, 21, and 23, we see that the proposed FGRS provides superior results to UPPRS. Comparing performance with different sequences, the PSNR improvement of FGRS over UPPRS is the most significant in sequence Akiyo, followed by sequence Foreman and Stefan. Sequence Stefan is the most challenging one with the most complex scene and the highest motion. The source coding rates of the FGS enhancement layer bitstream of Akiyo, Foreman, and Stefan are 354.69 kbps, 747.74 kbps, and 975.70 kbps. Hence, given the same amount of bits allocated by FGRS, the PSNR of sequence Stefan is the smallest among the three. Considering the gain in the Y component, FGRS yields 0.76 dB to 1.38 dB improvement compared to UPPRS as shown in Table 1.

To validate the performance of the proposed algorithm, the performance in terms of the overall PSNR of the Y components at various wireless channel conditions is shown in Figure 25, where we consider a two-state Markov model at various speeds and SNRs [29]. Figure 25a shows the 3D plots of the overall PSNR. At all wireless channel conditions, FGRS outperforms UPPRS.

Figure 25b shows the overall PSNR at various speeds at SNR = 10 dB. Fixed SNR value gives the same BER of the wireless channel. The higher the speed is, the more bursty the bit error of the wireless channel is. In other words, the larger the transition probability is. From the results, we see that the PSNR drops as the speed increases. The higher the transition probability is, the higher the packet loss rate is, given

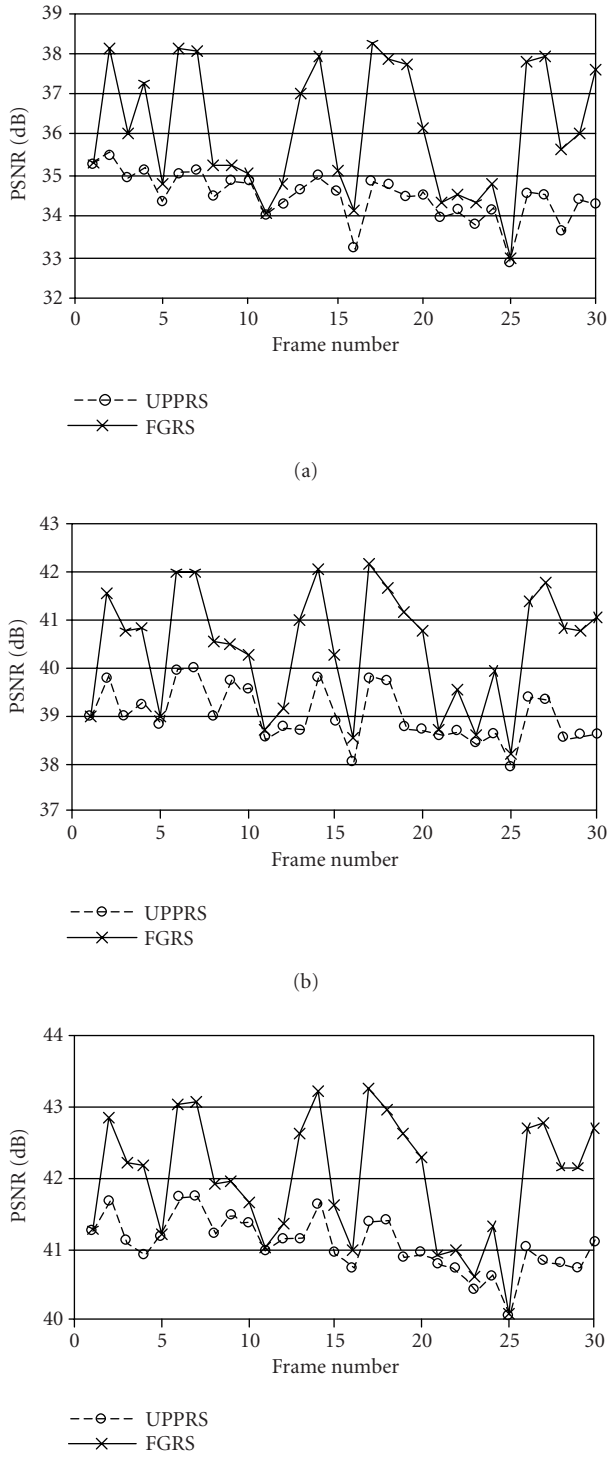


Figure 19: Frame-by-frame PSNR of UPPRS and FGRS with sequence Akiyo: (a) PSNR of the Y component, (b) PSNR of the U component, and (c) PSNR of the V component.

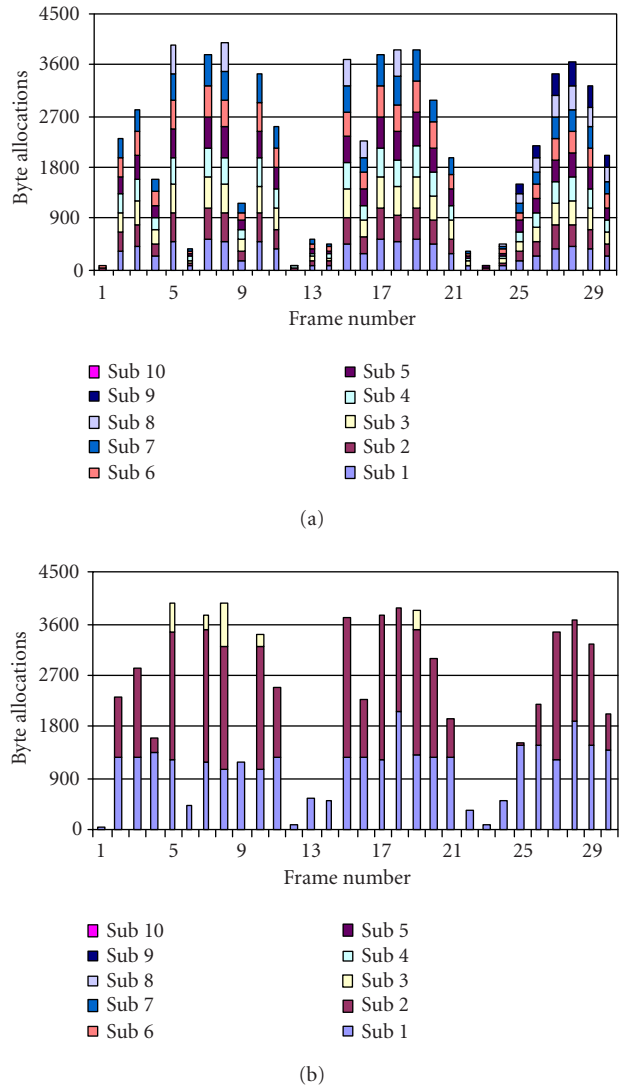


Figure 20: Sublayer byte allocations with sequence Foreman by (a) UPPRS and (b) FGRS.

the same BER. Higher packet loss rate has the effect of requiring more parity bits in the shaped bitstream, and higher probability of corrupting the packets that carries the shaped bitstream, thus, the PSNR value is lower.

Figure 25c shows the overall PSNR at various SNRs at speed = 10 km/h. Fixed speed gives the same burstiness of the bit errors of the wireless channel. The larger the SNR is, the smaller the BER is. We see from the results that the PSNR value increases with SNR. Smaller packet loss rate then leads to a higher PSNR.

Optimization for video streaming needs to be real time. As mentioned, in the training process for the model-based hypersurface, only a few number of operational RD states need to be examined, which saves the time. Thus, the two-stage RD optimization is preferred over the hill-climbing-based approach. In addition, as mentioned in Section 3.2, Step 2 can be skipped without too much performance degradation.

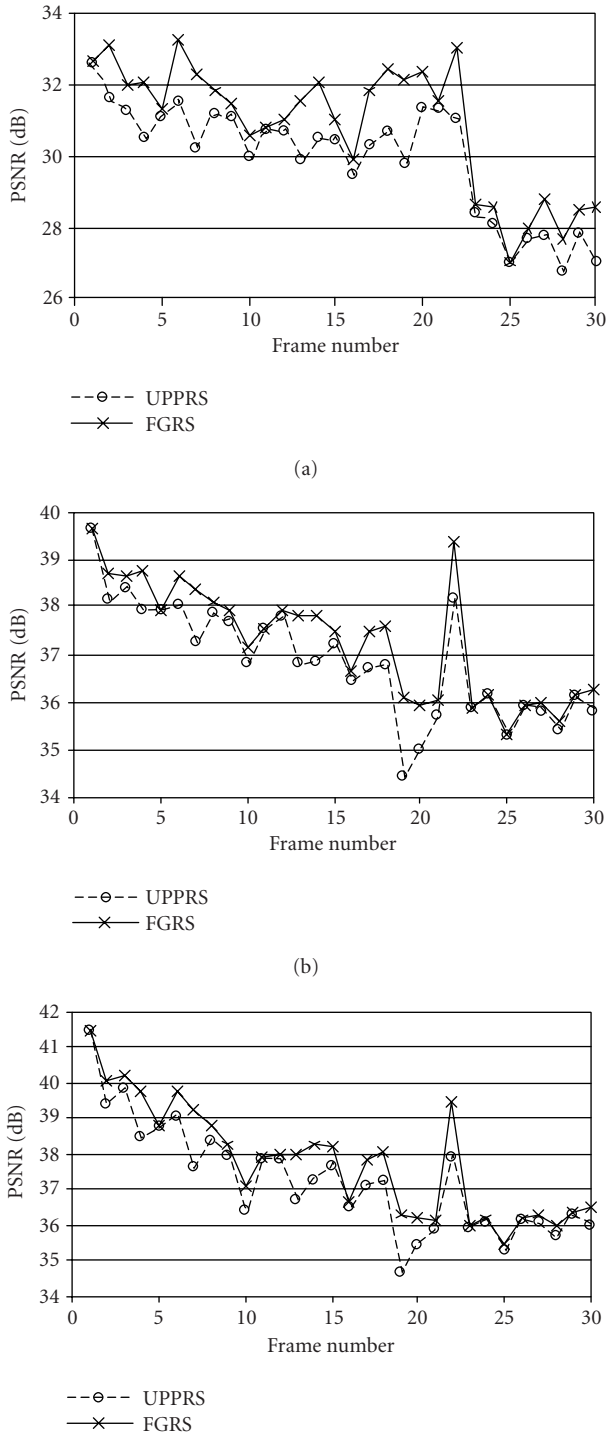


FIGURE 21: Frame-by-frame PSNR of UPPRS and FGRS with sequence Foreman: (a) PSNR of the Y component, (b) PSNR of the U component, and (c) PSNR of the V component.

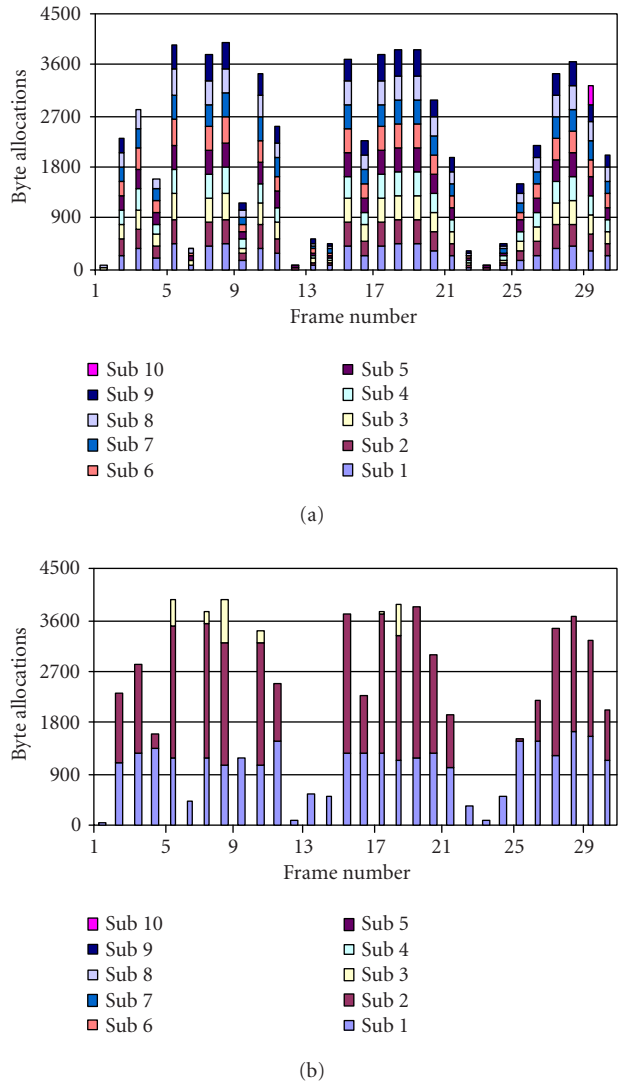
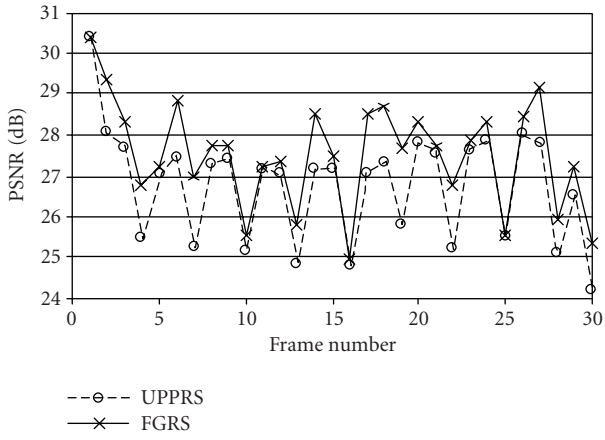


FIGURE 22: Sublayer byte allocations with sequence Stefan by (a) UPPRS and (b) FGRS.

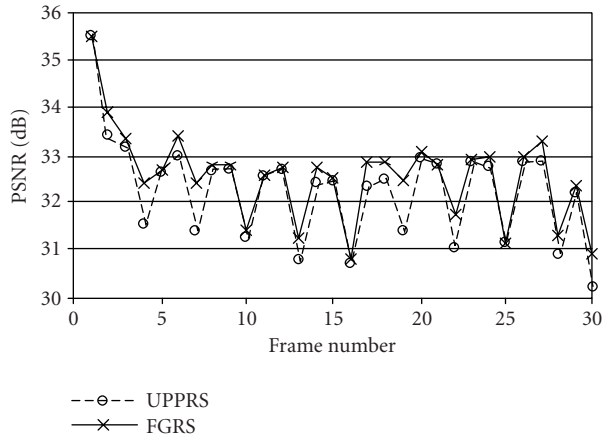
5. CONCLUSION

We proposed in this paper a novel FGRS approach to perform bandwidth adaptation for the precoded video, which is both FGS coded and FEC coded. FGRS utilizes the fine granularity property of FGS and FEC. Moreover, FGRS optimizes video streaming rather than achieves heuristic objectives. A two-stage rate-distortion (RD) optimization algorithm is used. The two-stage RD optimization algorithm finds the solution efficiently. The proposed FGRS outperforms UPPRS.

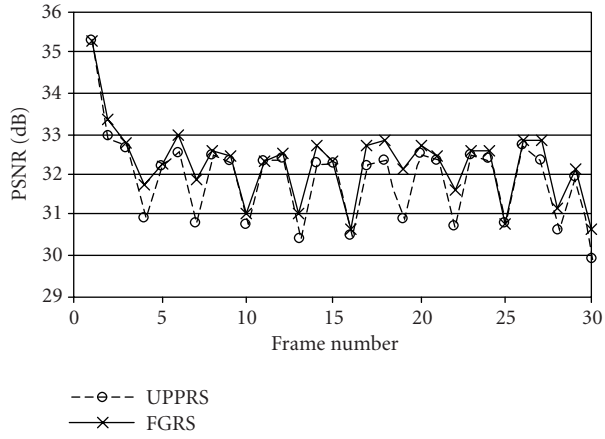
The novelty of the paper lies in three aspects. Although FGS has been proposed to provide fine granularity for precompressed video, none of the prior works has shown how to adapt the rate of the FGS bitstream that is protected by the FEC codes. Note that related work performs FEC encoding for the FGS bitstream at the time of delivery. Secondly, we formulate the FGRS problem as an RD optimization problem, while the work by van der Schaar and Radha [20] is not optimized but to achieve a certain target recovery rate. In addition, the distortion measure, which is called



(a)

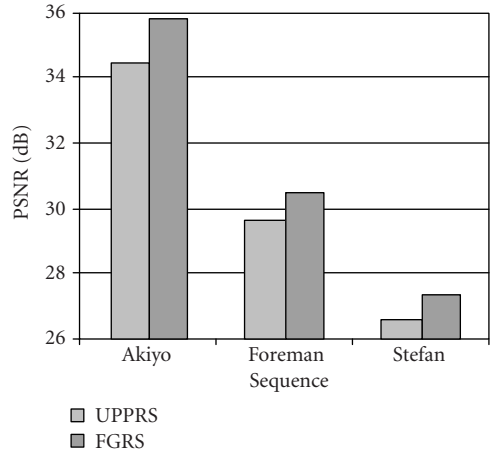


(b)

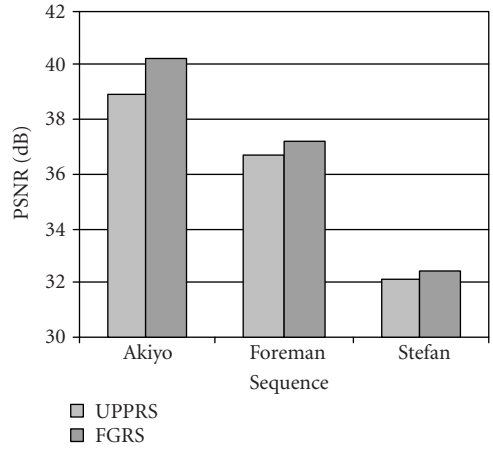


(c)

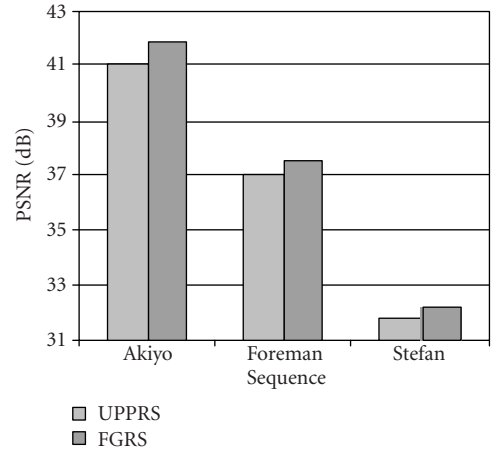
Figure 23: Frame-by-frame PSNR of UPPRS and FGRS with sequence Stefan: (a) PSNR of the Y component, (b) PSNR of the U component, and (c) PSNR of the V component.



(a)

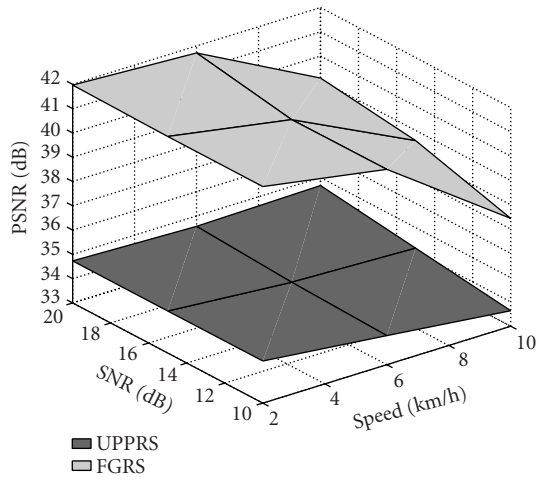


(b)

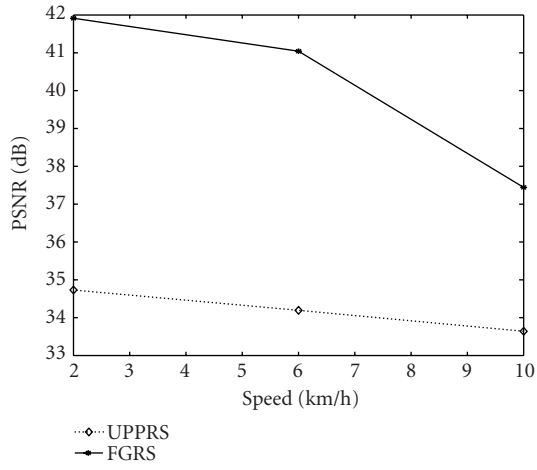


(c)

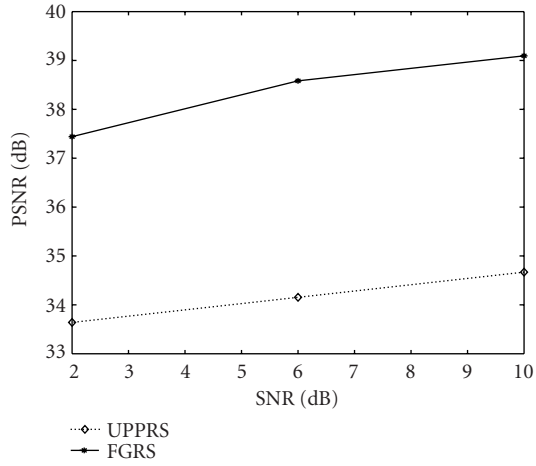
Figure 24: Overall PSNR of UPPRS and FGRS with sequences Akiyo, Foreman, and Stefan: (a) PSNR of the Y component, (b) PSNR of the U component, and (c) PSNR of the V component.



(a)



(b)



(c)

Figure 25: Performance (PSNR of the Y component) of all methods at various wireless channel conditions for sequence Foreman: (a) 3D view of PSNR at various speeds and SNRs; (b) PSNR at various speeds; (c) PSNR at various SNRs.

```
For (i = 1; i <= h; i++)
    If r_i < k_i
        a = k_i − r_i
        //the difference needed to satisfy r_i > k_i
        b = c = 0
        For (j = h; j >= 1 & c < a; j − −)
            b = r_j > a ? a : r_j
            //the symbols got from Sublayer j
            c+ = b
            r_j − = b
        End
        r_i = k_i
    End
End
```

Algorithm 2: Pseudocodes satisfying $r_i \geq k_i$ after bit allocation.

"gain" in the paper, is derived from the current packet loss rate in addition to the video characteristics. The gain is defined as the expected gain given the current packet loss rate. Prior work of DRS defines the distortion measure solely from the video characteristics. Thirdly, the RD optimization problem is solved by the proposed two-stage RD optimization algorithm, which can achieve the optimal solution fast. It is crucial that optimization for video streaming is done in real time.

Future work includes considering the smoothness criterion in FGRS optimization such as [33] to smooth the fluctuating PSNR resulted from the time-varying network conditions. Such fluctuation is not inherent from the FGRS algorithm. We can also investigate more the effect of outdated network information on FGRS, in addition to the simulation done in this paper by delaying the network bandwidth feedback. Moreover, deploying FGRS in a large network system, such as the "end system multicast (ESM)" [34] system, can be an exciting future research direction.

ACKNOWLEDGMENTS

This work was supported in part by Industrial Technology Research Institute. The authors would like to acknowledge the suggestions of Professor Mihaela van der Schaar, University of California at Davis, Professor Jose Moura and Professor Rohit Negi, Carnegie Mellon University, Professor Alex Eleftheriadis and Professor Shih-Fu Chang, Columbia University, Professor Antonio Ortega, University of Southern California, and the reviewers of the paper.

REFERENCES

[1] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, 2000.

[2] J. Cabrera, A. Ortega, and J. I. Ronda, "Stochastic rate-control of video coders for wireless channels," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 496–510, 2002.

[3] Z. He, J. Cai, and C. W. Chen, "Joint source channel rate-distortion analysis for adaptive mode selection and rate control in wireless video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 511–523, 2002.

[4] G. Cheung and A. Zakhor, "Bit allocation for joint source/channel coding of scalable video," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 340–356, 2000.

[5] L. P. Kondi, F. Ishtiaq, and A. K. Katsaggelos, "Joint source-channel coding for motion-compensated DCT-based SNR scalable video," *IEEE Trans. Image Process.*, vol. 11, no. 9, pp. 1043–1052, 2002.

[6] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in *Proc. 5th International Workshop on Networking and Operating System Support for Digital Audio and Video*, pp. 95–106, Durham, NH, USA, April 1995.

[7] W. Zeng and B. Liu, "Rate shaping by block dropping for transmission of MPEG-precoded video over channels of dynamic bandwidth," in *Proc. 4th ACM International Conference on Multimedia*, pp. 385–393, Boston, Mass, USA, November 1996.

[8] S. Jacobs and A. Eleftheriadis, "Streaming video using dynamic rate shaping and TCP congestion control," *Journal of Visual Communication and Image Representation*, vol. 9, no. 3, pp. 211–222, 1998.

[9] S. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, Englewood Cliffs, NJ, USA, 1995.

[10] T. P.-C. Chen and T. Chen, "Adaptive joint source-channel coding using rate shaping," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Orlando, Fla, USA, May 2002.

[11] D. S. Turaga and T. Chen, "Fundamentals of video compression: H.263 as an example," in *Compressed Video over Networks*, M.-T. Sun and A. R. Reibman, Eds., pp. 3–34, Marcel Dekker, NY, USA, 2001.

[12] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman & Hall, NY, USA, 1997.

[13] Motion Pictures Experts Group, "Overview of the MPEG-4 standard," ISO/IEC JTC 1/SC 29/WG 11 N 2459, 1998.

[14] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Communications*, vol. 36, no. 4, pp. 389–400, 1988.

[15] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, 2001.

[16] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.

[17] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, 1998.

[18] H. M. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, 2001.

[19] M. van der Schaar and H. M. Radha, "A hybrid temporal-SNR fine-granular scalability for internet video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 318–331, 2001.

[20] M. van der Schaar and H. M. Radha, "Unequal packet loss resilience for fine-granular-scalability video," *IEEE Trans. Multimedia*, vol. 3, no. 4, pp. 381–394, 2001.

[21] X. K. Yang, C. Zhu, Z. G. Li, G. N. Feng, S. Wu, and N. Ling, "A degressive error protection algorithm for MPEG-4 FGS video streaming," in *Proc. 2002 IEEE International Conference on Image Processing*, pp. 737–740, Rochester, NY, USA, September 2002.

[22] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 819–828, 2000.

[23] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," submitted to IEEE Trans. Multimedia.

[24] J. Chakareski, P. A. Chou, and B. Aazhang, "Computing rate-distortion optimized policies for streaming media to wireless clients," in *Proc. Data Compression Conference*, Snowbird, Utah, USA, April 2002.

[25] G. Cote, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error-prone networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 952–965, 2000.

[26] J. Postel, "User datagram protocol (RFC 768)," Internet Engineering Task Force, Internet draft, RFC-768, August, 1980, http://www.ietf.org/rfc/rfc768.txt.

[27] L.-A. Larzon, M. Degermark, and S. Pink, "Efficient use of wireless bandwidth for multimedia applications," in *Proc. 6th IEEE International Workshop on Mobile Multimedia Communications*, pp. 187–193, San Diego, Calif, USA, November 1999.

[28] A. J. Ganesh, "Estimating effective bandwidths from traffic data," in *Proc. IEEE Conference on Global Communications*, pp. 654–658, London, UK, November 1996.

[29] J.-P. Ebert and A. Willig, "A Gilbert-Elliot bit error model and the efficient use in packet level simulation," Tech. Rep. TKN-99-002, Telecommunication Networks Group, Technical University of Berlin, Berlin, 1999.

[30] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in *18th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 345–352, NY, USA, March 1999.

[31] S. A. Khayam, S. S. Karande, M. Krappel, and H. M. Radha, "Cross-layer protocol design for real-time multimedia applications over 802.11b networks," in *IEEE 2003 International Conference on Multimedia and Expo*, pp. 425–428, Baltimore, Md, USA, July 2003.

[32] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "An end-to-end TCP-friendly streaming protocol for multimedia over wireless internet," in *IEEE 2003 International Conference on Multimedia and Expo*, pp. 429–432, Baltimore, Md, USA, July 2003.

[33] X. M. Zhang, A. Vetro, Y. Q. Shi, and H. Sun, "Constant quality constrained rate allocation for FGS coded video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 121–130, 2003.

[34] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, 2002.

Trista Pei-chun Chen received her B.S. and M.S. degrees from National Tsing Hua University, Hsinchu, Taiwan, in 1997 and 1999, respectively, and her Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 2003. Trista is currently a video architect at NVIDIA Corporation, Santa Clara, California, performing design and testing of video hardware. From July 1998 to June 1999, she was a software engineer developing fingerprint identification algorithms at Startek Engineering Incorporated, Hsinchu, Taiwan. During the summer of 2000, she was with

HP Cambridge Research Laboratory, Cambridge, Massachusetts, conducting a research in image retrieval for massive databases. During the summer of 2001, she was with Pittsburgh Sony Design Center, Pittsburgh, Pennsylvania, designing circuits for video watermarking (VWM). Her research interests include multimedia hardware, networked video, watermark/data hiding, image processing, and biometric signal processing. She is a Member of the IEEE.

Tsuhan Chen has been with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania since October 1997, where he is now a Professor. He directs the Advanced Multimedia Processing Laboratory. His research interests include multimedia signal processing and communication, audio-visual interaction, biometrics, processing of 2D/3D graphics, bioinformatics, and building collaborative virtual environments. From August 1993 to October 1997, he worked in the Visual Communications Research Department, AT&T Bell Laboratories, Holmdel, New Jersey, and later at AT&T Labs-Research, Red Bank, New Jersey. Tsuhan helped create the Technical Committee on Multimedia Signal Processing, as the Founding Chair, and the Multimedia Signal Processing Workshop, both in the IEEE Signal Processing Society. He has recently been appointed as the Editor-in-Chief for IEEE Transactions on Multimedia for 2002–2004. He has coedited a book, *Advances in Multimedia: Systems, Standards, and Networks*. Tsuhan received the B.S. degree in electrical engineering from the National Taiwan University in 1987, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, California, in 1990 and 1993, respectively. He is a recipient of the National Science Foundation CAREER Award.

EURASIP Journal on Applied Signal Processing 2004:2, 192–206

SMART: An Efficient, Scalable, and Robust Streaming Video System

Feng Wu
Microsoft Research Asia, 3F Sigma Center, No. 49 Zhichun Road, Haidian, Beijing 100080, China
Email: fengwu@microsoft.com

Honghui Sun
Microsoft Research Asia, 3F Sigma Center, No. 49 Zhichun Road, Haidian, Beijing 100080, China
Email: hongsun@microsoft.com

Guobin Shen
Microsoft Research Asia, 3F Sigma Center, No. 49 Zhichun Road, Haidian, Beijing 100080, China
Email: jackysh@microsoft.com

Shipeng Li
Microsoft Research Asia, 3F Sigma Center, No. 49 Zhichun Road, Haidian, Beijing 100080, China
Email: spli@microsoft.com

Ya-Qin Zhang
Microsoft Research Asia, 3F Sigma Center, No. 49 Zhichun Road, Haidian, Beijing 100080, China
Email: yzhang@microsoft.com

Bruce Lin
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA
Email: blin@microsoft.com

Ming-Chieh Lee
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA
Email: mingcl@microsoft.com

Received 9 December 2002; Revised 12 September 2003

SMART, the acronym of *scalable media adaptation and robust transport*, is a suite of compression and transmission technologies for efficient, scalable, adaptive, and robust video streaming over the best-effort Internet. It consists of two indispensable parts: SMART video coding and SMART video streaming. The SMART video coding part is an efficient DCT-based universal fine granularity scalable coding scheme. Since the SMART video coding scheme adopts multiple-loop prediction and drifting reduction techniques at the macroblock level, it can achieve high coding efficiency at a wide range of bit rates. More importantly, it provides all sorts of scalabilities, that is, quality, temporal, spatial, and complexity scalabilities, in order to accommodate heterogeneous time-variant networks and different devices. The SMART video streaming part is a transport scheme that fully takes advantages of the special features of the scalable bitstreams. An accurate bandwidth estimation method is first discussed as the prerequisite of network adaptation. Then, flexible error resilience technique and unequal error protection strategy are investigated to enhance the robustness of streaming SMART bitstream. The SMART system shows excellent performances with regard to high coding efficiency, flexible channel bandwidth adaptation, smooth playback, and superior error robustness in static and dynamic experiments.

Keywords and phrases: video streaming, fine granularity scalability, video transmission, bandwidth estimation, error resilience, unequal error protection.

1. INTRODUCTION

With the recent developments in computing technology, compression and transmission technologies, high-capacity storage devices, and high-speed wired and wireless networks, more and more users expect to enjoy high-quality multimedia services over the Internet [1, 2, 3]. In general, there are two approaches to provide multimedia services on demand: *offline downloading* and *online streaming*. Since the streaming approach enables users to experience a multimedia presentation on the fly while it is being downloaded from the Internet, it has prevailed in both the academia and the industry. In virtue of the streaming techniques, users no longer have to suffer from long and even unacceptable transport time for full download.

Figure 1 exemplifies a typical scenario for streaming the same content to users. Raw video sequences are usually compressed in advance and then saved in the storage device. Upon the client's request, the streaming server retrieves compressed bitstream from the storage device and delivers it through the Internet that consists of many heterogeneous subnetworks. Receivers may use different devices for decoding, presenting the received video data at different resolutions, different frame rates, and different qualities depending on their connection speeds and device capabilities.

In fact, such multimedia streaming services create several challenges which may lie in technical fields even beyond video compression. These challenges mainly include but are not limited to the following.

(1) *Contents*

Multimedia contents are huge and growing rapidly. For example, only from RealNetworks Company's statistics in 2001 [4], over 350 000 hours of live sports, music, news, and entertainment contents were transmitted over the Internet every week. Furthermore, there are several hundred thousand hours of contents available on demand. To efficiently and effectively deliver such huge multimedia contents, advanced compression and transmission technologies are crucial.

(2) *Networks*

The networks used to deliver multimedia contents are becoming more and more complicated and heterogeneous. Additionally, unlike traditional dedicated networks, since the general best-effort Internet lacks quality of service (QOS) guarantee, network conditions themselves may be changing from time to time. This requires that compressed multimedia contents are deliverable over different networks from narrowband to broadband and from wired to wireless networks. It also requires that the delivery mechanism is able to adapt to network variations while providing a consistent user experience. In addition, since packet loss or channel error is inevitable during transmission, advanced error control technologies are required to protect the transmitted data.

(3) *Devices*

End-user devices are also becoming very different in processing power, memory, display resolution, and bandwidth. This requires tailoring multimedia contents and delivery schemes to best fit each device in order to provide the best possible multimedia user experience.

A straightforward solution would be to independently compress the same video sequence into many nonscalable bitstreams for every possible bit rate, frame rate, resolution, and device complexity. Actually, this solution has been extensively applied to most of the commercial streaming products, such as Windows Media Player system and Real Player system [4, 5]. When a video sequence is retrieved, the streaming server chooses an appropriate version of bitstream according to actual connection speed and device capability, and then transmits it to the user.

Obviously, video streaming systems based on the nonscalable compression techniques have several problems in taking the above challenges. Firstly, nonscalable video bitstreams are not able to adapt to time-variant networks. Even though switching among multiple nonscalable bitstreams is allowed at some key frames that are either compressed without prediction or coded with an extra lossless coded switching bitstream, such streaming systems only provide coarse and sluggish capability in adapting to bandwidth variations due to limitation in both the number of bitstreams and the number of key frames. Some studies have tried to solve this problem by switching at a special predictive frame, for example, S frame in [6], SP frame in [7], and SF frame in [8], which can reduce switching overhead and provide more switching points at the same cost. Secondly, nonscalable video bitstream is very sensitive to transmitted errors because almost every bit in the bitstream is very important and indispensable for decoding a group of pictures (GOP).

On the other hand, the scalable media adaptation and robust transport (SMART) system proposed in this paper is based on scalable compression techniques and is able to provide efficient, adaptive, and robust video streaming over the Internet. The core of the system is an efficient and universal fine granularity scalable (FGS) video codec. It uses multiple versions of references with increasing quality to make motion prediction more accurate for improved coding efficiency. At the same time, a drifting reduction technique is proposed to prevent possible error propagation due to corrupted high-quality references. When the two techniques are applied at the macroblock level, the SMART system can achieve a good trade-off between low drifting errors and high coding efficiency. Besides efficient fine granularity quality scalability, the SMART system supports efficient temporal and spatial scalabilities by utilizing similar techniques. Furthermore, the fine granularity scalability on complexity is also achieved by adjusting the decoding resolution, frame rate, and bit rate. In fact, the SMART system provides a universal scalable coding framework. For a sequence, the generated bitstreams can be served to a vast range of applications from low bit rate to high bit rate and from a PC device to a non-PC device without complicated transcoding.

The SMART video streaming part is a transport scheme that fully takes advantage of the special features of SMART video bitstream. It first estimates the available channel bandwidth through a hybrid model-based and probe-based

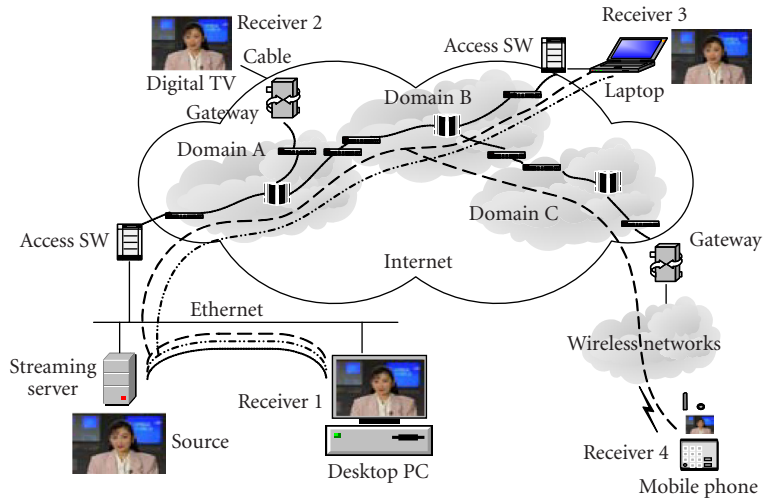


Figure 1: An exemplified scenario for streaming video.

method. Afterward, the transmitted video bitstreams are truncated to a bit rate that fits well in the estimated channel bandwidth. Since packet losses are inevitable in the general Internet, error control mechanism is a key component in this part. A flexible error resilience technique is proposed to adaptively enhance the robustness of SMART video bitstream. In addition, the SMART system provides a layered bitstream structure with a more important base layer and less important enhancement layers. Forward error correction (FEC) and automatic retransmission request (ARQ) techniques are applied to the base layer so as to reduce packet loss ratio and retransmission delay.

The rest of this paper is arranged as follows. Section 2 gives a brief overview of the SMART system. The SMART video coding techniques are discussed in Section 3. Section 4 introduces the channel estimation method used in the SMART system. The flexible error resilience technique and unequal error protection are described in Section 5. The experimental results presented in Section 6 demonstrate the advantages of the SMART system. Finally, Section 7 concludes this paper.

2. OVERVIEW OF THE SMART SYSTEM

This section gives an overview of the SMART coding and streaming system. At present, there are two modes for a streaming server to deliver video data to users: *multicast* and *unicast*. In the multicast mode, the server needs to send only one bitstream to a group of users, which is automatically replicated to all group members [9, 10], but this requests that the network has to be equipped with multicast-enable routers. In the unicast mode, the server delivers video bitstream to each user individually. The connection conditions between the server and each user can be estimated and monitored during transmission.

Since many routes in the current Internet do not enable the multicast mode, the SMART system discussed in this paper will focus on the unicast applications. Figure 2 illustrates the block diagram of the SMART system. Source video is first input into the SMART encoder module to generate a base layer bitstream and one or two enhancement layer bitstreams. Besides bitstreams, the SMART encoder generates a description file for each enhancement bitstream that contains all information for flexible error resilience and packetization. The detailed coding techniques will be discussed in Section 3, and the description file is introduced in Section 5. If the SMART encoder is powerful enough for real-time compression, the generated bitstreams can be directly packed and delivered just as in the live streaming applications. For the on-demand streaming applications, both the generated bitstreams and description files are saved in the storage device for future retrieval.

When the user submits a request to the SMART streaming server, like the *real-time streaming protocol* (RTSP) [11], the retrieved content, destination address, and user device capability are first transmitted by the transmission control protocol (TCP). After the control module in the SMART server receives the request, one user datagram protocol (UDP) connection is established immediately between the server and the user. Both the video data and the feedback from the SMART client are transmitted by this UDP connection. At the same time, the control module informs the server to retrieve the requested content from the storage device.

In the initial stage, the SMART system does not know the current channel conditions between the server and the client. Thus the base layer bitstream is packed with the *real-time transport protocol* (RTTP) [12] format using default channel parameters. At the same time, a prespecified FEC strategy is used in the base layer bitstream to generate parity packets. In general, since the base layer bit rate is very low in the SMART system, several seconds of source and parity packets can be rapidly delivered to the client as prebuffering. By transmitting these packets, the statistic channel parameters, such as packet loss ratio and latency, are packed with the

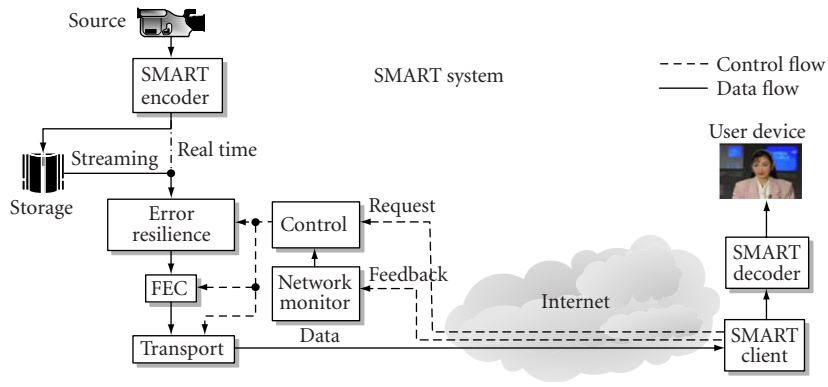


Figure 2: The block diagram of the SMART system.

real-time control protocol (RTCP) format [12] and sent back to the network monitor module in the SMART server. Accordingly, the SMART server can estimate the current available channel bandwidth.

With the obtained channel parameters, the SMART server starts to optimally pack the base layer and enhancement layer bitstreams with RTTP format. FEC protection depth to the base layer can be also adaptive to the channel conditions. In order to avoid network congestion, the actual bandwidth for the enhancement layer is the remaining part of the estimated channel bandwidth after delivering the base layer and FEC packets. Since the enhancement layer bitstream provides bit level scalability, it can be readily and precisely truncated to fit in the given bandwidth. Consequently, the SMART system can fully utilize available channel bandwidth and provide the user with better quality. Packet loss ratio and latency are periodically sent back by the client. The SMART server can timely adjust data transmission according to the feedbacks and the estimated channel bandwidth.

In the SMART system, another important feedback from the client is the negative acknowledgement (NACK) to notify the SMART server in which base layer packets are lost during transmission. Since the base layer is still a nonscalable bitstream, any lost packet would make the quality of the frames followed in the same GOP degrade rapidly. Therefore, the ARQ technique is also used to protect the base layer in the SMART system. Once the client detects lost packets at the base layer, a feedback is immediately sent out. The server will rapidly retransmit the lost packets. At the same time, any ARQ request received by the server will affect the sending rate to prevent further congestion in the channel. Since the base layer bit rate is very low in the SMART system, they can be strongly protected with small overhead bits. In addition, SMART video coding also provides the enhancement layer with an inherent error recovery feature. Any lost packet does not cause obvious visual artifacts. Moreover, it can be gracefully recovered in the following frames. Therefore, the current SMART system does not have any protection to the enhancement layer bitstreams.

In the following sections, the key techniques used in the SMART system, such as SMART video coding, bandwidth estimation, error resilience, and unequal error protection, will be discussed in detail.

3. SMART VIDEO CODING

How to efficiently compress video data with various scalabilities of rate, quality, temporal, spatial, and complexity is an active research topic in video coding field. Scalable video coding techniques have been developed rapidly in the past decade. Among them, spatial and temporal scalable coding techniques that provide video presentation at different resolutions, and frame rates have been accepted in some main video coding standards such as MPEG-2, MPEG-4, and H.263++ [13, 14, 15].

In addition, FGS video coding techniques have been extensively studied in recent years. MPEG-4 standard already accepted the bit plane coding technique in the streaming video profile (SVP) [16, 17]. In MPEG-4 FGS, an encoder using the motion-compensated discrete cosine transformation (DCT) transform coding generates a base layer video as the lowest quality layer. The residue between the original image and the reconstructed base layer image forms the enhancement layer with the bit plane coding technique, which provides an embedded bitstream and fine granularity quality and temporal scalabilities.

One major feature in MPEG-4 FGS is that the base layer and all the bit planes at the enhancement layer in a predicted frame are always compensated from the reconstructed version of the base layer in the reference. Therefore, it provides a remarkable capability in both bandwidth adaptation and error recovery. By predicting the enhancement layer from the base layer, any bitstream truncation and lost packets at the enhancement layer have no effect on the frames followed. However, this also makes MPEG-4 FGS suffer from severe degradation in coding efficiency due to the lowest quality reference. Furthermore, it is difficult for MPEG-4 FGS to compress different-resolution video at different layers; otherwise,

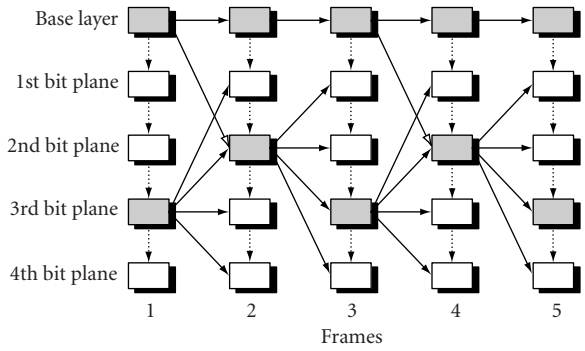


Figure 3: The proposed multiple-loop prediction technique with two references case.

the coding efficiency at the enhancement layer would be further degraded.

Therefore, the SMART video coding is proposed based on our previous works [18, 19]. The multiple-loop prediction and drifting reduction techniques are first used at the quality enhancement layer to achieve a good trade-off between high coding efficiency and low drifting errors. Then, these techniques are extended to the temporal and spatial scalabilities, consequently, forming an efficient and universal scalable video coding framework.

3.1. Multiple-loop prediction

The multiple-loop prediction technique was first proposed in [18, 19] to improve the coding efficiency of MPEG-4 FGS. The basic idea is to use as many predictions from the enhancement layer as possible instead of always using the base layer as in MPEG-4 FGS. Because the quality of a frame is higher at the enhancement layer than at the base layer, this will make motion prediction more accurate, thus improving the coding efficiency. Considering the cost by introducing multiple references at the enhancement layer, Figure 3 illustrates a typical multiple-loop prediction scheme with one additional reference used in the enhancement layer coding.

In Figure 3, the gray rectangular boxes denote the reconstructed base layer or the reconstructed enhancement layer at a certain bit plane as references for the next frame coding. Solid arrows with solid lines between two adjacent frames are for temporal prediction, solid arrows with dashed lines are for prediction in the transform domain, and hollow arrows with solid lines are for reconstruction of high-quality reference from the previous base layer. Each frame at the base layer is always predicted from the previous frame at the base layer (low-quality reference) so as to avoid any effect from the lost enhancement data. Each frame at the enhancement layer is predicted from the previous frame at the enhancement layer (high-quality reference) for high coding efficiency.

In the FGS video coding schemes, the base layer bit rate is usually very low. It is reasonable to assume that the base layer bitstream can be completely transmitted to the client. Since the base layer is still predicted from the previous base layer, any bitstream truncation and lost packets at the enhancement layer have no effect on the base layer video. However, when those bit planes used to reconstruct the high-quality reference are truncated or corrupted during transmission, this would inevitably cause drifting errors at the enhancement layer. As a result, the decoded enhancement layer video may be deteriorated rapidly.

3.2. Drifting reduction

In order to effectively reduce drifting errors at the enhancement layer, the basic idea is to make sure that the encoder and the decoder have the same reconstructed reference for any future frame prediction, although the reconstructed reference may not have the best quality it could get if reconstructed using the high-quality reference.

We will show this idea through an example in Figure 3. In the decoder end, if the third bit plane in Frame 1 is truncated or dropped which is used in the encoder end to get the high-quality reference, the enhancement layer in Frame 2 will have to use the previous low-quality reference instead. Of course, some quality losses would be introduced by doing so. However, as long as in both the encoder end and the decoder end the reconstruction of the high-quality reference of Frame 2 always uses the base layer of Frame 1 as the reference, then the errors in Frame 1 could not further propagate to any frames followed. In other words, the reference used for prediction could be different from that used for reconstruction. This feature will prevent the errors drifting and preserve all the bandwidth adaptation and error recovery features as in MPEG-4 FGS.

As shown by hollow arrows with solid lines in Figure 3, some frames, such as Frames 2 and 4, reconstruct the high-quality references from the previous low-quality reference at both the encoder and the decoder to prevent the errors propagating into future frames. However, if the third bit plane of Frame 1 is available at the decoder end, a better second bit plane quality of Frame 2 can still be reconstructed from the high-quality reference for display purpose only. In other words, the reconstruction of display image can be different from that of reference image.

Although the proposed technique significantly reduces the drifting errors from the previous frames, it still has a negative effect on coding efficiency because the high-quality reference does not always get the best quality it could get. If the reference for prediction and reconstruction is chosen as frame-based, that is, all enhancement layer macroblocks in a frame with the same reference, it is very difficult for the SMART video coding to provide a good trade-off between high coding efficiency and low drifting errors.

3.3. Macroblock-based mode selection

The technique choosing the proper reference for prediction and reconstruction at each enhancement layer macroblock is first proposed in [20]. Derived from MPEG-4 FGS and Figure 3, three intercoding modes as shown in Figure 4 are defined for coding the enhancement inter macroblock. The rectangular boxes in the first row denote the base layer and the rectangular boxes in other rows denote bit planes at the

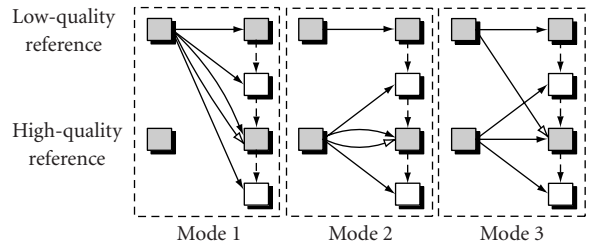


Figure 4: Three intercoding modes for the quality enhancement layer.

enhancement layer. Gray rectangular boxes indicate those to be reconstructed as references. Solid arrows with solid lines between two adjacent frames are for temporal predictions, solid arrows with dashed lines are for prediction in the transform domain, and hollow arrows with solid lines are for reconstruction of high-quality reference from the previous base layer.

In Mode 1, the base layer and the enhancement layer are both predicted and reconstructed from the previous low-quality reference. Since the low-quality reference is always available at the decoder, there is no drifting error in this mode. The coding efficiency of this mode is low due to low-quality temporal prediction. If all enhancement layer macroblocks are encoded with this mode, the proposed scheme is similar to MPEG-4 FGS.

In Mode 2, the base layer is predicted and reconstructed from the previous low-quality reference, but the enhancement layer is predicted and reconstructed from the previous high-quality reference. It can significantly improve the coding efficiency at moderate and high bit rates. There is no drifting error at the base layer. When the channel bandwidth is not high enough to transmit the high-quality reference, this mode would cause drifting errors at the enhancement layer.

In Mode 3, the enhancement layer is predicted from the previous high-quality reference while reconstructed from the previous low-quality reference at both the encoder and the decoder. This mode was for the purpose of drifting reduction. Since the low-quality reference is always consistent at both the encoder and the decoder, the drifting errors propagated from previous high-quality references can be eliminated with Mode 3.

More intercoding modes could be readily added in the SMART coding as long as they have the virtue in improving coding efficiency or reducing error propagation. In order to achieve a good trade-off between low drifting errors and high coding efficiency, a mode selection algorithm is proposed to choose the proper coding mode for each macroblock. Besides the above three intermodes, intramode is allowed in the enhancement layer coding. Intramode or intermode is determined by motion estimation. If a macroblock is encoded with the intramode at the base layer, the corresponding enhancement macroblock is also encoded with the intramode without temporal prediction. If a macroblock at the base layer is encoded with temporal prediction, the proposed mode selection algorithm has to determine which intercoding mode should be used at the corresponding enhancement macroblock.

The reference for prediction in Mode 1 is of low quality but the reference used in Mode 2 and Mode 3 is of high quality. If the absolute mean of the predicted DCT residues produced in Mode 1 is less than that in Modes 2 and 3, the current macroblock is coded using Mode 1; otherwise, the mode selection algorithm further determines the coding mode between Mode 2 and Mode 3. Both Modes 2 and 3 are predicted from the high-quality reference, the difference between them lies in the reference for reconstruction. In general, most of the enhancement macroblocks should be coded with Mode 2 for high coding efficiency. Mode 3 is used only when the drifting errors are more than a given threshold. In order to estimate the potential drifting errors at the encoder, the iterative drifting model proposed in [21] is given as follows:

$$y(n) = \begin{cases} 0, & n = 1, \\ \mathrm{MC}_n\left(y(n-1) + \mathrm{DCT}^{-1}\left(X(n-1)\right)\right), & N \geq n > 1. \end{cases} \tag{1}$$

Here, N is the total number of frames in a GOP, $\mathrm{MC}(\cdot)$ and DCT^{1} denote motion compensation and IDCT, respectively, $y(n-1)$ is the accumulative error propagated to the $(n-1)$th frame, and $X(n-1)$ is DCT coefficients encoded in those bit planes for reconstruction of the high-quality reference in the $(n-1)$th frame. With motion compensation, their sum forms the next drifting errors in the nth frame. If the estimated drifting error $y(n)$ is more than the given threshold, this macroblock is encoded with Mode 3; otherwise, this macroblock is encoded with Mode 2.

For the convenience of a better understanding of the proposed multiple-loop prediction, drifting reduction, and macroblock-based mode selection, Figure 5 illustrates an exemplified block diagram of the SMART decoder with quality scalability. There are two reference frames in the decoder. The first one is located in the base layer decoder and stored in the frame buffer 0 as a low-quality reference, while the second one is located in the enhancement layer decoder and stored in the frame buffer as a high-quality reference.

Only the low-quality reference is allowed in the reconstruction of the base layer in order to assure that no drifting error exists at this layer. The enhancement layer can use two different quality references for reconstruction. The enhancement bitstream is first decoded using bit plane variable length decoding (VLD) and mode VLD. The bit planes at the enhancement layer are categorized into a lower enhancement layer and a higher enhancement layer. Only the bit planes at the lower enhancement layer are used to reconstruct the high-quality reference. In Figure 5, $n(t)$ is the number of bit planes at the lower enhancement layer and $m(t)$ is the number of additional bit planes for the reconstruction of the display frame.

The decoded block-based bit planes are used to reconstruct the DCT coefficients of the lower and higher enhancement layers using the bit plane shift modules. After inverse

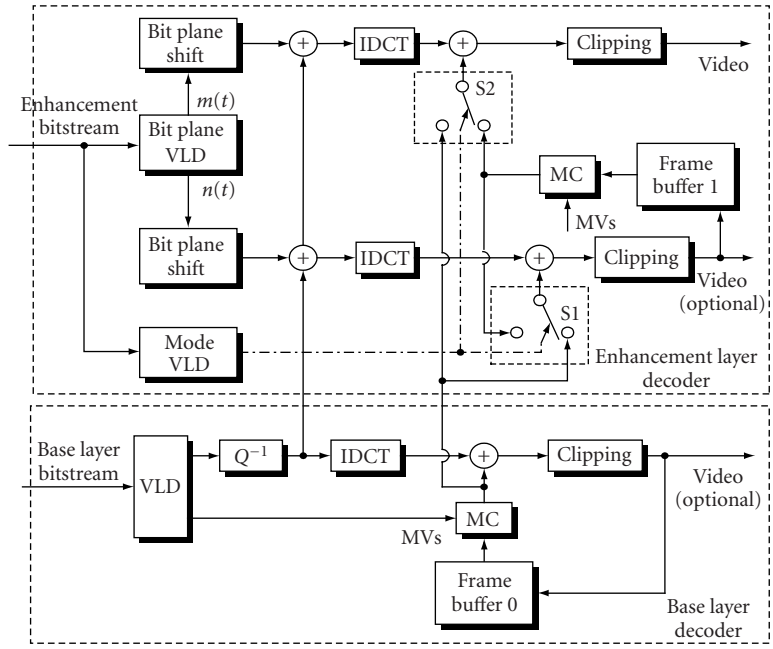


Figure 5: The exemplified SMART decoder with quality scalability.

DCT, the lower enhancement DCT coefficients plus the reconstructed base layer DCT coefficients generate the error image for reference, and all DCT coefficients including the higher enhancement layer generate the error image for display. Furthermore, there are two switches S1 and S2 at the SMART decoder that control which temporal prediction is used at each enhancement macroblock. The decoded macroblock coding mode decides the actions of the two switches. When one macroblock is coded as Mode 1, the switches S1 and S2 connect to the low-quality prediction. When it is coded as Mode 2, both of the switches S1 and S2 connect to the high-quality prediction. When it is coded as Mode 3, the switch S1 connects to the low-quality prediction. However, the switch S2 still connects to the high-quality prediction. Since the display frame does not cause any error propagation, the display frame is always reconstructed from the high-quality prediction in Mode 3.

3.4. Universal scalable coding framework

The techniques discussed in Sections 3.1, 3.2, and 3.3 can be readily extended to the temporal and spatial scalable video coding. The basic idea is to use more than one enhancement layer based on a common base layer to implement fine granularity quality, temporal, and spatial scalabilities within the same framework. In order to achieve high coding efficiency for various scalabilities, multiple prediction loops with different quality references are employed in the proposed framework. For example, by utilizing the high-quality reference in the spatial enhancement layer coding, the proposed framework can likewise fulfill efficient spatial scalability. The complexity scalability is inseparable with other scalabilities in the SMART codec. It is achieved by increasing/decreasing the bit rate, frame rate, and resolution. The changes in the frame rate and resolution provide coarse scalability on complexity. Because of the property of fine granularity of each layer on bit rate, the SMART codec also provides fine scalability on complexity by adjusting the bit rate of each layer. The lowest complexity bound is the low-resolution base layer decoding, which should be sufficiently low for many applications.

Figure 6 illustrates the proposed universal scalable coding framework. Source video with two resolutions is compressed in the proposed framework. Narrow rectangles denote low-resolution video and wide rectangles denote high-resolution video. There are two different enhancement layers sharing a common base layer, and two optional enhancement ones. The bottom layer is the base layer. It is usually generated as the lowest quality, lowest resolution, least smoothness, and least complexity. The quality enhancement layer compresses the same resolution video as that at the base layer. It will improve the decoded quality of the base layer. The temporal enhancement layer improves the base layer frame rate and makes the decoded video look smooth. The rest two enhancement layers improve the video quality and frame rate at high resolution. These two enhancement layers are optional in the proposed framework and appear only if the video with two different resolutions is encoded. The same resolution enhancement layers are stored in the same bitstream file. Therefore, the SMART coding scheme generates at most three bitstreams: one base layer bitstream and two enhancement layer bitstreams.

Except that the base layer is encoded with the conventional DCT transform plus VLC technique, all of the enhancement layers are encoded with the bit plane coding technique. In other words, every enhancement layer bitstream can be arbitrarily truncated in the proposed framework. In

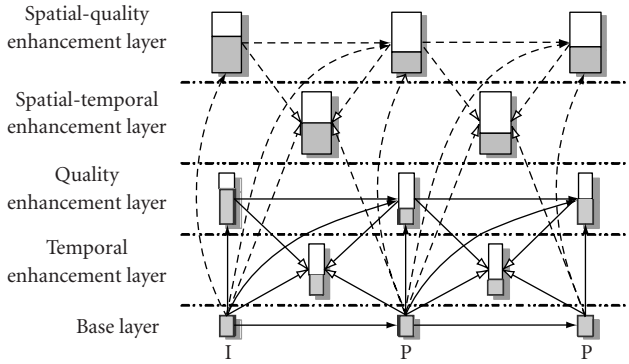


Figure 6: The proposed SMART coding framework.

Figure 6, each rectangle denotes the whole frame bitstream at one enhancement layer. The shadow region is the actual transmitted part, whereas the blank region is the truncated part. Hence the proposed SMART video coding provides the most flexible bit rate scalability.

Since the multiple-loop prediction technique is used in the proposed framework, every layer, excluding the base layer, can select the prediction from two different references. As shown by solid arrows with solid lines in Figure 6, the quality enhancement layer use the reconstructed base layer and the reconstructed quality enhancement layer at a certain bit plane as references. As shown by hollow arrows with solid lines, the temporal enhancement layer is bidirectionally predicted from the base layer and the quality enhancement layer. The predictions for the two high-resolution enhancement layers are denoted by solid arrows with dashed lines and hollow arrows with dashed lines, respectively.

Similarly, some intercoding modes are defined at the temporal and spatial enhancement layers, which can be found in [22, 23, 24]. Each coding mode has its unique references for prediction and reconstruction. The similar mode selection algorithm discussed in Section 3.3 can be also applied to the temporal and spatial enhancement layers. In fact, some other techniques proposed in [25, 26, 27, 28] can be easily incorporated into the framework by defining several new coding modes.

4. CHANNEL ESTIMATION

In the streaming applications, one important component is congestion control. Congestion control mechanisms usually contain two aspects: estimating channel bandwidth and regulating the rate of transmitted bitstream. Since the SMART video coding provides a set of embedded and full scalable bitstreams, rate regulation in the SMART system is essentially equal to truncating bitstreams to a given bit rate. There is not any complicated transcoding needed. The remaining problem is how to estimate the channel bandwidth.

Typically, channel estimation techniques are divided into two categories: probe-based and model-based. The probe-based techniques estimate the channel bandwidth bottleneck by adjusting the sending rate in a way that could maintain packet loss ratio below a certain threshold [29]. The model-based techniques are based on a TCP throughput model that explicitly estimates the sending rate as a function of recent packet loss ratio and latency. Specifically, the TCP throughput model is given by the following formula [30]:

$$\lambda = \frac{1.22 \times \mathrm{MTU}}{\mathrm{RTT} \times \sqrt{p}}, \tag{2}$$

where λ is the throughput of a TCP connection (in B/s), MTU is the packet size used by the connection (in bytes), RTT is the round-trip time of the connection (in seconds), and p is the packet loss ratio of the connection.

With formula (2), the server can estimate the available bandwidth by receiving feedback parameters RTT and p from the client.

Among all existing model-based approaches, TCP-friendly rate control (TCP-FRC) [31] is the most deployable and successful one. The sending rate formula, by considering the influence of time out, is given as follows:

$$\lambda = \frac{\mathrm{MTU}}{\mathrm{RTT}\sqrt{2p/3} + \mathrm{RTO}\left(3\sqrt{3p/8}\right)p\left(1+32p^2\right)}, \tag{3}$$

where RTO is the TCP retransmission time-out value (in seconds).

However, TCP-FRC has one obvious drawback undesirable for the SMART system, that is, the estimated bandwidth always fluctuates periodically even if the channel bandwidth is very stable. The reason is that TCP-FRC is trying to increase the sending rate when there is no lost packet. This unfortunately leads to a short-term congestion. Since TCP-FRC is very sensitive in the low packet loss ratio case, the sending rate is greatly reduced again to avoid further congestion.

Therefore, the SMART system adopts a hybrid model-based and probe-based method to estimate the available channel bandwidth. TCP-FRC is first used to calculate an initial estimated bandwidth by packet loss ratio and RTT. If there is no lost packet, the estimated bandwidth should be more than the previous estimation. On the other hand, some packets that contain less important enhancement data are transmitted with the probing method. This is a feature of the SMART bitstream. Even though those packets are lost for probing, they do not affect other data packets. In general, the estimated bandwidth by the probing method is viewed as the bottleneck between the server and the client. The estimated bandwidth in TCP-FRC should be not more than that estimated by the probing method. Therefore, the probing method provides an upper bound for TCP-FRC so as to reduce fluctuations in bandwidth estimation.

Video packets in the SMART system are categorized into three priorities for bandwidth allocation. The retransmitted and base layer packets have the highest priority. Estimated bandwidth is first used to deliver them to the client. The FEC packets of the base layer have the second priority. If the estimated bandwidth is more than that needed by the highest priority packets, they are delivered prior to the enhancement packets. Finally, the remaining channel bandwidth is

used to deliver the truncated enhancement bitstreams. In fact, the enhancement packets also implicates several different priorities, For example, the bit planes for reconstruction of the high-quality reference are more important than other bit planes, and at low bit rates, the quality enhancement layer may be more important than the temporal enhancement layer, and so on. Because of limitation in pages, this paper no longer further discusses this issue.

Frame:	17302	Bits:	0	Type:	2	Time 0:	19	Max layer:	9
VP start:	17808	Bits:	5	BP_num:	0	isGLL:	0	MB_num:	0
VP start:	17822	Bits:	3	BP_num:	0	isGLL:	0	MB_num:	1
VP start:	18324	Bits:	0	BP_num:	2	isGLL:	0	MB_num:	81

Figure 7: The exemplified description file.

5. ERROR CONTROL

In the streaming applications, error control mechanism is another important component to ensure received bitstreams decodable, which often includes error resilience, FEC, ARQ, and even error concealment [32, 33]. In this section, we will discuss the error resilience technique and unequal error protection used in the SMART system.

5.1. Flexible error resilience

Packet losses are often inevitable while transmitting compressed bitstreams over the Internet. Besides the necessary frame header, some resynchronization markers and related header information have to be inserted in the bitstream generation so that the lost packets do not affect other data packets. This is the most simple error resilience technique, but very useful. The resynchronization marker plus the header and data followed is known as a *slice*. In MPEG-4, the resynchronization marker is a variable length symbol from 17 bits to 23 bits [14]. The slice header only contains the index of the first macroblock in this slice. In general, the resynchronization marker and the slice header are inserted at a given length or number of macroblocks.

However, this method has two obvious problems when it is applied to the enhancement layer bitstream in the SMART system. Firstly, although the SMART enhancement layer bitstream provides bit level scalability, the actual minimum unit in the packetization process is a slice. This would greatly reduce the granularity of scalability. Secondly, the slice length is decided in the encoding process and fixed in the generated bitstream. For the streaming applications, it is impossible to adjust the slice length again to adapt to channel conditions. In general, longer slice means lower overhead bits and bigger effects of lost packet. On the contrary, shorter slice means higher overhead bits and lower effects of lost packet. Adaptively adjusting the slice length is a very desirable feature in the streaming applications. Therefore, a flexible error resilience technique is proposed in the SMART enhancement layer bitstream.

In the SMART system, there are no resynchronization markers and slice headers in the enhancement layer bitstream. Thus, the generated bitstream is exactly the same as that without error resilience. But the positions of some macroblocks and their related information needed in the slice header are recorded in a description file. Besides the index of the first macroblock, the slice header at the enhancement layer also contains the located bit plane of the first macroblock. We call these macroblocks *resynchronization points*. Note that each resynchronization point is always macroblock aligned. In this stage, resynchronization points do not cause actual overhead bits in the generated bitstreams. Thus, the description file could even record every macroblock.

Figure 7 exemplifies the structure of the description file. The fields *Frame* and *Bits* in the same row are used to locate the start position of a frame in the bitstream. The units of these two fields are byte and bit, respectively. The field *Bits* is always zero in the first row of every frame due to byte-aligned. The field *Type* indicates the frame type: 0 for I frame, 1 for P frame, and 2 for B frame. The field *time* is the relative time of the current frame. The first digit in this field denotes the number of seconds, and the second digit denotes the frame index in a second. The field *Max Layer* is the maximum number of bit planes in a frame. The fields *VP start* and *Bits* are used to locate the start position of a macroblock. The field *BP_num* is the located bit plane of the current macroblock. The field *isGLL* indicates whether this macroblock is used to reconstruct the high-quality reference or not. It provides a priority to transmit the enhancement bitstreams. The field *MB_num* is the first macroblock index in a slice.

The proposed flexible error resilience is used only at the enhancement DCT data. If the motion vectors exist at the enhancement layer, for example, in temporal frames, they are differentially coded together before DCT coefficients. The VOP header and coded motion vectors are processed as a slice. There is not any resynchronization point within them in case that the lost motion vectors in a slice affect other motion vectors decoded in another slice due to motion vector prediction. Similar to the entropy coding used in MPEG-4 FGS, there is not any DC and/or AC coefficient prediction among neighboring blocks. Therefore, the slices in a frame have no dependency except for the inherent relationship among bit planes.

With the description file, the proposed error resilience technique in the SMART system can choose any resynchronization points to chop an enhancement layer bitstream into slices. However, since the position of the resynchronization point may be not byte-aligned in the bitstream, one lost packet probably makes many packets followed undecodable. As showed in Figure 8, macroblock N is a resynchronization point. It shares byte m in the bitstream with macroblock $N-1$. If the macroblock N is selected as the start of a slice, these two macroblocks may not locate in the same transport packet. If byte m belongs to the previous packet, the packet of macroblock N is even received undecodable when the packet of macroblock $N-1$ is lost during transmission.

A simple technique is proposed to solve this problem as shown in Figure 8. When a resynchronization point is selected as the start of one slice, the first byte of this macroblock

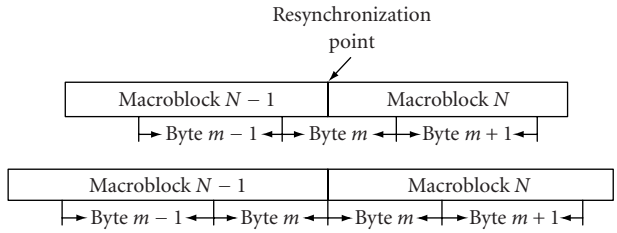


Figure 8: The error resilience in the SMART system.

is duplicated into two slices so that the lost packet cannot affect each other. This leads to the probability that the head and tail of each slice may have several useless bits. The decoder has to know how many useless bits should be skipped. Therefore, the numbers of useless bits in the head and tail generated from the description file need to be encapsulated into the transport packet and transmitted to the client. The fields *MB_num* and *BP_num* at the slice header also need to be encapsulated into the transport packet and transmitted to the client.

We evaluate the proposed error resilience technique compared with that in MPEG-4. In the proposed technique, a byte has to be duplicated for every selected resynchronization point. In addition, the corresponding numbers of useless bits are also contained in the packet. But, bits for the resynchronization marker in MPEG-4 bitstream can be saved. Therefore, the proposed technique has the similar overhead bits in each slice. However, it enables the SMART system to adaptively adjust the slice length according to rate-distortion optimization and channel conditions. This is a very desirable feature in the streaming applications.

5.2. Unequal error protection

Since the SMART video coding provides a layered bitstream structure with a more important base layer and less important enhancement layers, error protection techniques such as FEC and ARQ are unevenly applied to the base layer and the enhancement layer.

In general, if the streaming systems have no request on delay, FEC would not play an important role because the lost packets can be recovered by ARQ. In the SMART system, the bit rate of the base layer is very low and it may only occupy a small part of the total bit rate (usually less than 20%). When four data packets are protected by one FEC packet, the overhead for FEC is only about 5%. In return, if the lost packets take place randomly, most of them may be recovered by FEC. It will considerably reduce the system delay due to ARQ. Based on these considerations, the SMART system uses FEC as an option at the base layer if low delay is requested in some applications. It also provides a space to achieve a better tradeoff between ARQ delay and FEC overhead.

When FEC is enabled, the base layer packets are divided into many groups containing K source packets per group. Assume that $N - K$ parity packets will be produced with a Reed-Solomon codec. When these N packets are transmitted over the best-effort Internet, any received subset of K source or parity packets can be used to reconstruct the original K source packets. In the SMART system, K is often set as $N - 1$ in order to avoid too much overhead introduced by FEC. The target using FEC is mainly to recover occasional lost packet and reduce the delay caused by ARQ.

The base layer bitstream in the SMART system is a nonscalable one. Furthermore, the motion compensation technique is used in the base layer coding. Any lost packet will make the quality of the frames followed in a GOP degrade rapidly. Therefore, the ARQ technique is also applied to the base layer to handle burst packet losses. If the lost packets that cannot be recovered from FEC are detected at the base layer, a NACK feedback is immediately sent to the server. If no acknowledgement feedback is received, the transmitted base layer packets are saved in a special buffer. The SMART will get the lost base layer packets from the special buffer and retransmit them to the client until time out. If the base layer packets arrive too late or are not able to be recovered by FEC and ARQ, the SMART system will skip to the next GOP. In addition, the client periodically sends the acknowledgement feedback so that the server discards the received base layer packets from the special buffer.

From the discussions in Section 3, we know that the SMART video coding provides the embedded enhancement bitstreams. Any truncation and lost packets at the enhancement bitstream are allowed. It can be gracefully recovered by the drifting reduction technique. Therefore, no error protection techniques are applied to the enhancement packets in the current SMART system. In fact, consider that the lost packets in low bit planes used to reconstruct the high-quality reference may still have a big effect on maintaining high decoded quality. The techniques for partly protecting the enhancement layer packets should be further investigated.

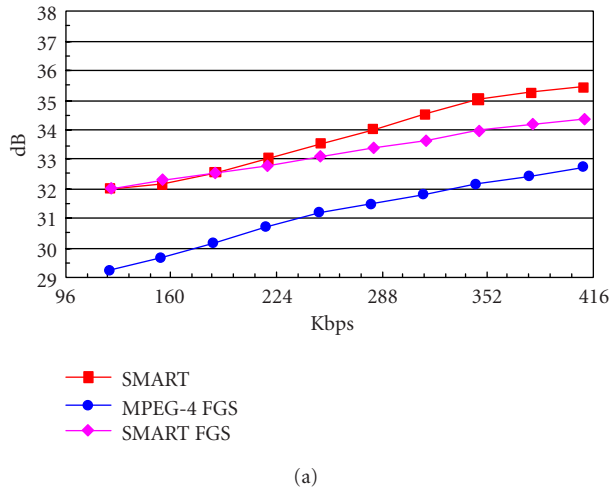
6. EXPERIMENTS

Both static and dynamic experiments are designed to evaluate the performances of the SMART system on coding efficiency, channel estimation, bandwidth adaptation, error robustness, and so on.

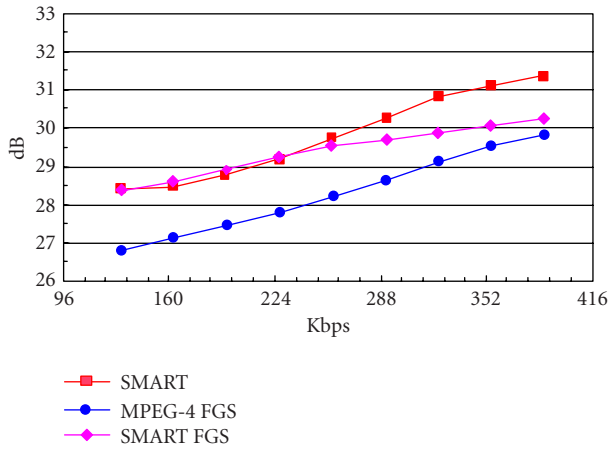
6.1. Static tests

Three different coding schemes, namely, MPEG-4 FGS without global motion compensation, SMART coding without multiple-loop prediction, and SMART coding, are compared in terms of coding efficiency. MPEG-4 FGS provides the reference of scalable coding scheme for comparisons. The final drift amendment (FDAM) software of MPEG-4 FGS released in June 2002 is used to create the results of MPEG-4 FGS [34]. The SMART system uses Windows Media Video Encoder 8.0 (WMV8) as the base layer codec. The MPEG-4 testing sequences Foreman and Coastguard with common intermediate format (CIF) are used in this experiment.

In the first set of experiments, the testing sequences are coded at 10 Hz encoding frame rate. Only the first frame is encoded as I frame and the rest of the frames are encoded as P frames. The main parameters in the MPEG-4 FGS base layer are given as follows:

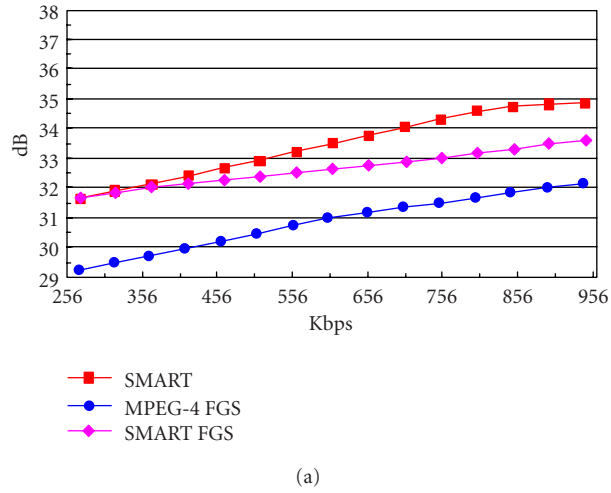


(a)

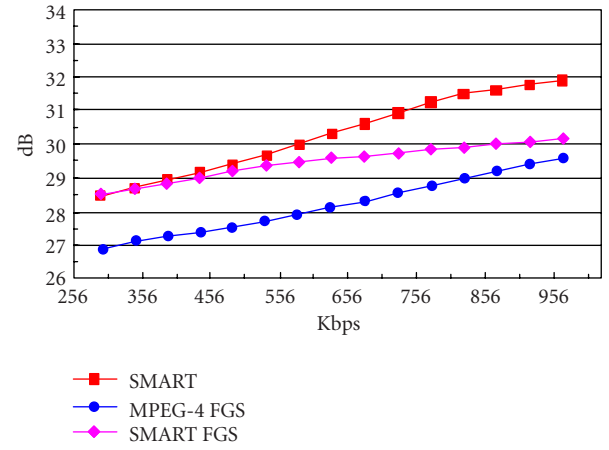


(b)

Figure 9: The curves of average PSNR versus bit rate at 10 fps without B frame and temporal scalability. (a) Foreman CIF Y (10 Hz). (b) Coastguard CIF Y (10 Hz).



(a)



(b)

Figure 10: The curves of average PSNR versus bit rate at 30 fps with B frame and temporal scalability. (a) Foreman CIF Y (30 Hz). (b) Coastguard CIF Y (30 Hz).

(i) motion estimation: ±32 pixels,
(ii) motion compensation: quarter-pixel precision,
(iii) quantization: MPEG,
(iv) direct search range: 2 (half-pixel unit),
(v) advanced prediction: Enable,
(vi) skipped macroblock: Enable.

The results of the first set of experiments are depicted in Figure 9. In the curves of MPEG-4 FGS, the base layer is coded with quantization parameter 31, and the quality enhancement layer bitstream is truncated at an interval of 32 kbps. By adjusting the quantization parameter, the SMART curve has a bit rate at the base layer similar to MPEG-4 FGS. The curves of SMART FGS are obtained with the SMART system by only using Mode 1. The curves of SMART are obtained with all the discussed coding techniques in this paper.

SMART FGS and SMART use the same coding technique at the base layer. Since only Mode 1 is used in SMART FGS, the enhancement layer coding is essentially the same as that in MPEG-4 FGS. WMV8 provides a very good base layer compared with MPEG-4; the coding efficiency gain at the base layer is close to 2.8 dB in Foreman and 1.6 dB in Coastguard compared with MPEG-4 FGS. But without the proposed enhancement prediction technique, the coding efficiency gain is becoming smaller and smaller with bit rates increasing. The coding efficiency gain of SMART FGS is only 1.6 dB in Foreman and 0.44 dB in Coastguard at the highest bit rate. However, the SMART curves with the proposed techniques present the consistent performance in a wide range of bit rates. The bit rate for the high-quality reference is about 346 kbps in Foreman and 322 kbps in Coastguard. The coding efficiency gain, when the high-quality reference is available, is 2.9 dB in Foreman and 1.7 dB in Coastguard.

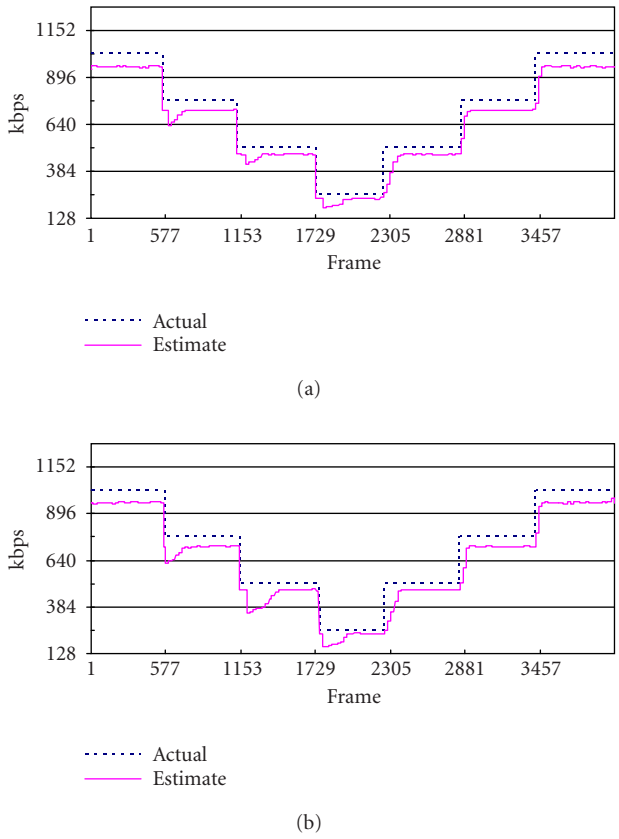


Figure 11: The estimated channel bandwidth in the SMART system. (a) Estimated bandwidth in bs_one sequence. (b) Estimated bandwidth in bs_two sequence.

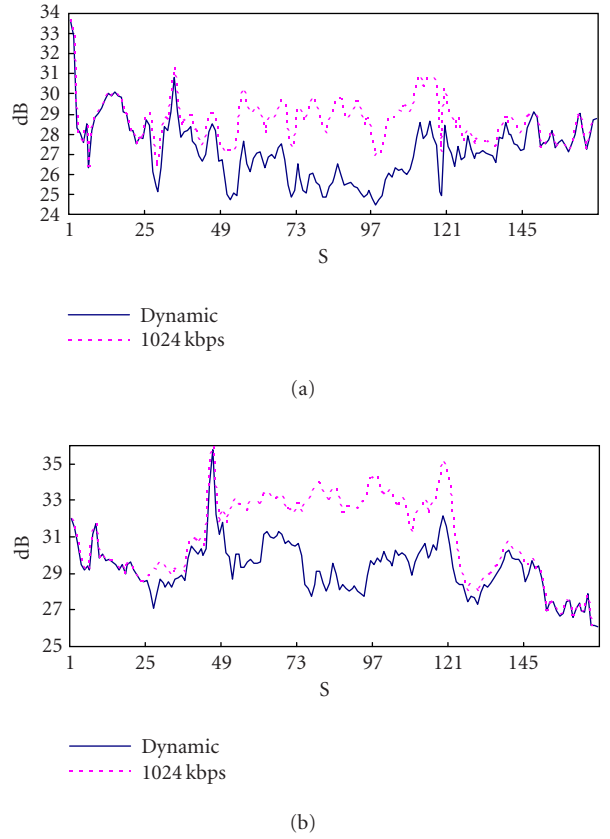


Figure 12: The decoded quality over the dynamic channel: (a) bs_one Y. (b) bs_two Y.

In addition, although the high-quality references are used in the enhancement layer coding, the SMART curves still have the similar performance as the SMART FGS curves at low bit rates. The SMART curve has only about 0.15 dB loss at 150 kbps. This proves that the proposed drifting reduction technique can effectively control the drifting errors.

In the second set of experiments, the testing sequences are coded at 30 Hz encoding frame rate. Only the first frame is coded as I frame. There are two temporal frames in the scalable coding scheme between a pair of I and P or two P frames. Other experimental conditions are the same as in the first set of experiments. The same experimental results given in Figure 10 are observed as in the first set of experiments.

Since neither MPEG-4 FGS nor the SMART codec contains one of the switching techniques, for example, S frame, SP frame, or SF frame, the readers who are interested in the comparisons between the scalable video coding and the SP frame on H.26L TML can read the MPEG proposal in [35].

6.2. Dynamic tests

The dynamic experiments try to test the SMART system under the dynamic channel, such as streaming video over the Internet, where the channel bandwidth varies in a wide range of bit rates. The conditions of MPEG-4 FGS verification test are used in this experiment [36]. Two CIF sequences, bs_one and bs_two, each with 4032 frames (168 seconds at 24 fps) are used. The channel bandwidth varies from 1024 kbps to 256 kbps and then recovers to 1024 kbps again with a step of 256 kbps. Every bit rate lasts 24 seconds. The dynamic channel simulation is done by the commerce simulator, the Cloud software (http://www.shunra.com).

By using the hybrid model-based and probe-based bandwidth estimation scheme, when the sequences bs_one and bs_two are transmitted over the simulated dynamic channel, the estimated bandwidth is recorded and plotted in Figure 11. The dashed-line curves are the actual channel bandwidth limited by the Cloud simulator. When the channel bandwidth switches from high bit rate to low bit rate, the estimated bandwidth with TCP-FRC can rapidly decrease in order to avoid network congestion. When the channel bandwidth increases, the estimated bandwidth can also catch this variation at a short time. Furthermore, the curves in Figure 11 fully demonstrate the advantage of the hybrid bandwidth estimation method, where the probing method gives an upper bound to prevent TCP-FRC from raising the sending rate over the network bottleneck. Therefore, the SMART system has a stable estimation when the channel bandwidth stays in a constant.

The decoded quality of sequences bs_one and bs_two are also recorded and plotted in Figure 12. Each sample is the average PSNR in a second. Two factors, channel bandwidth and video content, will affect the final decoded quality. Sometimes, even if the channel bandwidth is high, the decoded PSNR may not be high due to active content. In order to eliminate the video content factor in evaluating the performance of the SMART system on bandwidth adaptation, the PSNR curves decoded at 1024 kbps are drawn in Figure 12 as reference. The distances between the dynamic curve and the 1024 kbps curve reflect the bandwidth adaptation capability of the SMART system.

As shown in Figure 12, the decoded PSNR is less than that at 1024 kbps up to 4.4 dB from 73 to 96 seconds because the estimated bandwidth is only 240 kbps around. From 49 to 72 seconds and from 97 to 120 seconds, the estimated channel bandwidth is about 480 kbps. The decoded PSNR is significantly improved compared with that at 240 kbps. From 25 to 48 seconds and from 121 to 144 seconds, the estimated bandwidth is about 720 kbps. The decoded PSNR is only slightly less than that at 1024 kbps. The SMART system provides almost the same quality as that at 1024 kbps from 1 to 24 seconds and from 145 to 168 seconds. The estimated bandwidth in these two periods is about 950 kbps. Thus, the SMART system shows excellent performance on bandwidth adaptation.

Although there are a lot of packet losses while switching the channel bandwidth from high bit rate to low bit rate, with the proposed error resilience technique and unequal error protection, all packet losses at the base layer are recovered in the simulation. No green blocks appeared in the decoded video. For the enhancement bitstreams, there is not any error protection. The effects of packet losses at the enhancement layer are gradually recovered by the drifting reduction technique. There are also no obvious visual artifacts and quality degradation in the average PSNR curves.

At last, the SMART video player is given in Figure 13. It can real-time decode the CIF sequence at 1024 kbps with PIII 800 MHz. The decoded video is presented in the biggest window. The right-upper window shows the curve of the estimated channel bandwidth and the right-bottom window is for the program list. The packet loss ratio is drawn in the window between them. A progress bar is used to indicate the status of the received buffer.

The proposed SMART system is also used to run the results of MPEG-4 FGS verification tests, where the SMART codec is replaced by MPEG-4 FGS codec. The experimental results have been released in [37].

7. CONCLUSIONS AND FUTURE WORKS

The SMART system presents an efficient, adaptive, and robust scheme for streaming video over the Internet. Firstly, since the multiple-loop prediction and drifting reduction techniques are applied at the macroblock level, the SMART system can outperform MPEG-4 FGS up to 3.0 dB. Secondly, the SMART system has excellent capability in network bandwidth and device adaptation due to the embedded enhancement bitstreams and the universal scalabilities. Thirdly, with the proposed bandwidth estimation method, the SMART system can rapidly and stably catch bandwidth variations. At last, since a layered bitstream structure with a more important base layer and less important enhancement layers is provided in the SMART system, the base layer bitstream is highly protected by the proposed error resilience and unequal error protection techniques with small overhead. The SMART system can provide users with much smooth playback experience and much better visual quality in the best-effort Internet.

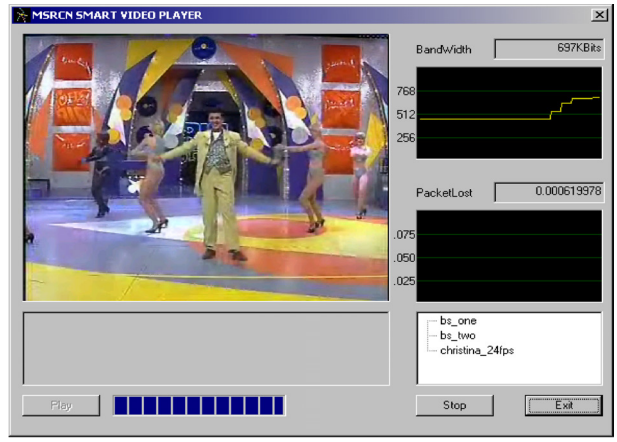


Figure 13: The interface of the SMART video player.

Although the SMART system shows good performances on coding efficiency, bandwidth adaptation, channel estimation, and error robustness, there are still several problems needed to be further studied in the future, such as how to further improve the coding efficiency to cover an even wider bit rate range; how to optimally allocate the available bandwidth to different enhancement layers so that the perception quality looks better; how to optimally packetize the base layer and the enhancement layer bitstreams so that the packet losses have less effects; how to optimally decide the parameters in FEC and ARQ to achieve a better trade-off between ARQ delay and FEC overhead; and how to protect those bit planes for reconstruction of the high-quality reference at the enhancement layers with small overhead. In addition, how to effectively utilize the features and techniques of the SMART system in the multicast applications is another topic worthy of further study.

ACKNOWLEDGMENTS

Many colleagues and visiting students in Microsoft Research Asia also took part in the SMART system. The authors would like to thank Dr. W. Zhu, Dr. Q. Zhang, and L. Wang for their contribution in the bandwidth estimation part; X. Sun for fine-grain quality and temporal scalability; Dr. Q. Wang and Dr. R. Yang for fine-grain spatial scalability; and Prof. Z. Xiong and S. Cheng for setting up the SMART server in Texas A&M University.

REFERENCES

[1] J. Lu, "Signal processing for Internet video streaming: a review," in *Proc. SPIE: Image and Video Communications and Processing 2000*, vol. 3974, pp. 246–259, San Jose, Calif, USA, January 2000.

[2] A. Luthra, *Need for simple streaming video profile*, ISO/IEC JTC1/SC29/WG11, M5800, Noordwijkerhout, The Netherlands, March 2000.

[3] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the Internet: approaches and directions," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, 2001.

[4] *RealNetworks facts*, 2001, http://www.realnetworks.com/company.

[5] *Windows Media technologies*, http://www.microsoft.com/windows/windowsmedia.

[6] N. Farber and B. Girod, "Robust H.263 compatible video transmission for mobile access to video servers," in *Proc. International Conference on Image Processing*, vol. 2, pp. 73–76, Santa Barbara, Calif, USA, October 1997.

[7] M. Jarczewicz and R. Kurceren, *A proposal for SP-frames*, ITU-T Q.6/SG 16, VCEG-L27, Elysee, Germany, January 2001.

[8] X. Sun, S. Li, F. Wu, G. B. Shen, and W. Gao, "Efficient and flexible drift-free video bitstream switching at predictive frames," in *Proc. IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 541–544, Lausanne, Switzerland, August 2002.

[9] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (ACM SIGCOMM '96)*, pp. 117–130, Stanford, Calif, USA, August 1996.

[10] D.-N. Yang, W. Liao, and Y.-T. Lin, "MQ: an integrated mechanism for multimedia multicasting," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 82–97, 2001.

[11] H. Schulzrinne, A. Rao, and R. Lanphier, *Real time streaming protocol (RTSP)*, Internet Engineering Task Force, Internet draft, RFC 2326, April 1998.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A transport protocol for real-time applications*, Internet Engineering Task Force, Internet draft, RFC 1889, January 1996.

[13] MPEG video group, *Information technology—Generic coding of moving pictures and associated audio*, ISO/IEC 13818-2, International standard, 1995.

[14] MPEG video group, *Generic coding of audio-visual objects: part 2*, ISO/IEC 14496-2, International standard, 1998.

[15] ITU-T Recommendation H.263, *Video coding for low bit rate communication*, Version 2, 1998.

[16] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, 2001.

[17] M. van der Schaar and H. Radha, "A hybrid temporal-SNR fine-granular scalability for Internet video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 318–331, 2001.

[18] F. Wu, S. Li, and Y.-Q. Zhang, "DCT-prediction based progressive fine granularity Scalability coding," in *Proc. International Conference on Image Processing (ICIP '00)*, vol. 3, pp. 566–569, Vancouver, British Columbia, Canada, September 2000.

[19] F. Wu, S. Li, and Y.-Q. Zhang, "A framework for efficient progressive fine granularity scalable video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 332–344, 2001.

[20] X. Sun, F. Wu, S. Li, W. Gao, and Y.-Q. Zhang, "Macroblock-based progressive fine granularity scalable video coding," in *Proc. IEEE International Conference on Multimedia and Expo (ICME '01)*, pp. 461–464, Tokyo, Japan, August 2001.

[21] F. Wu, S. Li, B. Zeng, and Y.-Q. Zhang, "Drifting reduction in progressive fine granular scalable video coding," in *Proc. Picture Coding Symposium*, Seoul, Korea, April 2001.

[22] X. Sun, F. Wu, S. Li, W. Gao, and Y.-Q. Zhang, "Macroblock-based progressive fine granularity scalable (PFGS) video coding with flexible temporal-SNR scalablilities," in *Proc. International Conference on Image Processing*, pp. 1025–1028, Thessaloniki, Greece, October 2001.

[23] Q. Wang, F. Wu, S. Li, Y. Zhong, and Y.-Q. Zhang, "Fine-granularity spatially scalable video coding," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 1801–1804, Salt Lake City, Utah, USA, May 2001.

[24] R. Yan, F. Wu, S. Li, R. Tao, and Y. Wang, "Efficient video coding with hybrid spatial and fine-grain SNR scalabilities," in *Proc. SPIE: Visual Communications and Image Processing 2002*, vol. 4671, pp. 850–859, San Jose, Calif, USA, January 2002.

[25] R. Kalluri and M. van der Schaar, *Single-loop motion-compensated based fine-granular scalability (MC-FGS) with cross-checked results*, ISO/IEC JTC1/SC29/WG11, M6831, Pisa, Italy, 2001.

[26] A. Reibman, L. Bottou, and A. Basso, "DCT-based scalable video coding with drift," in *Proc. International Conference on Image Processing*, pp. 989–992, Thessaloniki, Greece, October 2001.

[27] A. Reibman and L. Bottou, "Managing drift in DCT-based scalable video coding," in *Proc. IEEE Data Compression Conference*, pp. 351–360, Salt Lake City, Utah, USA, April 2001.

[28] W.-H. Peng and Y. K. Chen, "Mode-adaptive fine granularity scalability," in *Proc. International Conference on Image Processing*, pp. 993–996, Thessaloniki, Greece, October 2001.

[29] D. Wu, Y. T. Hou, W. Zhu, et al., "On end-to-end architecture for transporting MPEG-4 video over the Internet," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, no. 6, pp. 923–941, 2000.

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

[31] M. Handley, S. Floyd, J. Padhye, and J. Widmer, *TCP friendly rate control (TFRC): Protocol specification*, Internet Engineering Task Force, Internet draft, RFC 3448, January 2003.

[32] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 819–828, 2000.

[33] P. A. Chou and Z. Miao, "Rate-distortion optimized sender-driven streaming over best-effort networks," in *Proc. IEEE 4th Workshop on Multimedia Signal Processing*, pp. 587–592, Cannes, France, October 2001.

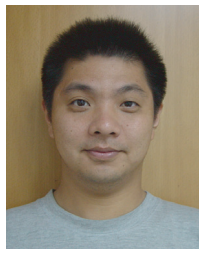
[34] Video group, *Information technology—Coding of audio-visual objects part 5, Amendment 1: Reference software for MPEG-4*, ISO/IEC JTC1/SC29/WG11, MPEG M4711, Jeju, March 2002.

[35] F. Wu, X. Sun, and S. Li, *Comparisons between PFGS and JVT SP*, ISO/IEC JTC1/SC29/WG11, MPEG m8426, Fairfax, 2002.

[36] Test group, *MPEG-4 visual fine granularity scalability tools verification test plan*, ISO/IEC JTC1/SC29/WG11, MPEG N4456, Pattaya, Thailand, 2001.

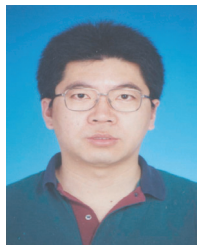
[37] Test group, *Report on MPEG-4 visual fine granularity scalability tools verification test*, ISO/IEC JTC1/SC29/WG11, MPEG N4791, Fairfax, 2002.

Feng Wu received his B.S. degree in electrical engineering from the University of Xi'an Electrical Science and Technology, Xi'an, China, in 1992, and his M.S. and Ph.D. degrees in computer science from Harbin Institute of Technology, Harbin, China, in 1996 and 1999, respectively. He joined Microsoft Research Asia, Beijing, China, as an Associate Researcher in 1999 and was promoted to a Researcher in 2001. He has played a major role in Internet Media Group in developing scalable video coding and streaming technologies. He has authored and coauthored over 60 papers in video compression and contributed some technologies to MPEG-4 and H.264. His research interests include video and audio compression, multimedia transmission, and video segmentation.



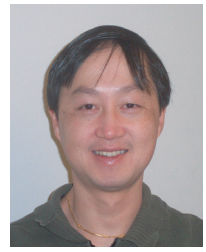
Honghui Sun received his B.S. degree from Zhejiang University, Hang Zhou, China, in 1992, and his M.S. degree in computer graphics from Beijing University, Beijing, China, in 1995, all in computer science. He was a Lecturer in Computer Science Department, Beijing University, Beijing, China, from 1995 to 1999. He joined Microsoft Research Asia, Beijing, China as a Research Software Design Engineer in 1999 and was promoted to Senior Research Software Design Engineer in 2001. His work mainly focuses on video compression, multimedia transmission, and network technology.

Guobin Shen received his B.S. degree from Harbin University of Engineering, Harbin, China, in 1994, his M.S. degree from Southeast University, Nanjing, China, in 1997, and his Ph.D. degree from the Hong Kong University of Science and Technology (HKUST) in 2001, all in electrical engineering. He was a Research Assistant at HKUST from 1997 to 2001. Since then, he has been with Microsoft Research Asia. His research interests include digital image and video signal processing, video coding and streaming, peer-to-peer networking, and parallel computing.



Shipeng Li received his B.S. and M.S. degrees from the University of Science and Technology of China (USTC) in 1988 and 1991, respectively, and the Ph.D. degree from Lehigh University, Bethlehem, PA, in 1996, all in electrical engineering. He was with the Electrical Engineering Department, University of Science and Technology of China, Hefei, China, from 1991 to 1992. He was a member of the technical staff at Sarnoff Corporation, Princeton, NJ, from 1996 to 1999. He has been a Researcher with Microsoft Research China, Beijing, since May 1999. His research interests include image/video compression and communications, digital television, multimedia, and wireless communication. He has contributed some technologies to MPEG-4 and H.264.

Ya-Qin Zhang received the B.S. and M.S. degrees in electrical engineering from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 1983 and 1985, and the Ph.D. degree in electrical engineering from George Washington University, Washington, DC, in 1989. He is currently the Managing Director of Microsoft Research Asia, Beijing, China, in 1999. He has authored and coauthored over 200 refereed papers in leading international conferences and journals, and has been granted over 40 US patents in digital video, Internet, multimedia, wireless, and satellite communications. Dr. Zhang served as Editor-in-Chief for the IEEE Trans. on Circuits and Systems for Video Technology from July 1997 to July 1999. He was the Chairman of the Visual Signal Processing and Communications Technical Committee of the IEEE Circuits and Systems (CAS) Society. He has received numerous awards, including several industry technical achievement awards and IEEE awards, such as the CAS Jubilee Golden Medal. He recently received the Outstanding Young Electrical Engineer of 1998 Award.



Bruce Lin received his B.S. degree from National Taiwan University in 1988 and his M.S. and Ph.D. degrees from the University of Maryland, College Park, in 1994 and 1996, respectively, all in computer science. He was a Research Assistant at the center for automatic research at the University of Maryland from 1992 to 1995. Since 1995, he has been working with Microsoft on video compression. Currently, he is a Development Manager of Media Processing Technology group in Microsoft Digital Media Division. His focus is on Windows media video and various image/video processing components for Windows.

Ming-Chieh Lee was born in Taiwan. He received his B.S. degree in electrical engineering from the National Taiwan University, Taiwan, in 1988, and his M.S. and Ph.D. degrees in electrical engineering from California Institute of Technology, Pasadena, in 1991 and 1993, respectively. His Ph.D. research topic was on still and moving image compression using multiscale techniques. From January 1993 to December 1993, he was with the Jet Propulsion Laboratory as a member of the technical staff and was working on multiresolution image transmission and enhancement. In December 1993, he joined the advanced video compression group of Microsoft Corporation, Redmond, Wash, as a Software Design Engineer. He is now the Product Unit Manager in charge of the Core Media Processing Technology group in Microsoft Digital Media Division. His group has produced technologies including Windows media video, Windows media audio, Windows media audio Professional, and Windows media audio voice.

EURASIP Journal on Applied Signal Processing 2004:2, 207–219

Optimal Erasure Protection Assignment for Scalable Compressed Data with Small Channel Packets and Short Channel Codewords

Johnson Thie
School of Electrical Engineering & Telecommunications, The University of New South Wales, Sydney, NSW 2052, Australia
Email: j.thie@ee.unsw.edu.au

David Taubman
School of Electrical Engineering & Telecommunications, The University of New South Wales, Sydney, NSW 2052, Australia
Email: d.taubman@unsw.edu.au

Received 24 December 2002; Revised 7 July 2003

We are concerned with the efficient transmission of scalable compressed data over lossy communication channels. Recent works have proposed several strategies for assigning optimal code redundancies to elements in a scalable data stream under the assumption that all elements are encoded onto a common group of network packets. When the size of the data to be encoded becomes large in comparison to the size of the network packets, such schemes require very long channel codes with high computational complexity. In networks with high loss, small packets are generally more desirable than long packets. This paper proposes a robust strategy for optimally assigning elements of the scalable data to clusters of packets, subject to constraints on packet size and code complexity. Given a packet cluster arrangement, the scheme then assigns optimal code redundancies to the source elements subject to a constraint on transmission length. Experimental results show that the proposed strategy can outperform previously proposed code redundancy assignment policies subject to the above-mentioned constraints, particularly at high channel loss rates.

Keywords and phrases: unequal error protection, scalable compression, priority encoding transmission, image transmission.

1. INTRODUCTION

In this paper, we are concerned with reliable transmission of scalable data over lossy communication channels. For the last decade, scalable compression techniques have been widely explored. These include image compression schemes, such as the embedded zerotree wavelet (EZW) [1] and set partitioning in hierarchical trees (SPIHT) [2] algorithms and, most recently, the JPEG2000 [3] image compression standard. Scalable video compression has also been an active area of research, which has recently led to MPEG-4 fine granularity scalability (FGS) [4]. An important property of a scalable data stream is that a portion of the data stream can be discarded or corrupted by a lossy communication channel without compromising the usefulness of the more important portions. A scalable data stream is generally made up of several elements with various dependencies such that the loss of a single element might render some or all of the subsequent elements useless but not the preceding elements.

For the present work, we focus our attention on "erasure" channels. An erasure channel is one whose data, prior to transmission, is partitioned into a sequence of symbols, each of which either arrives at the destination without error, or is entirely lost. The erasure channel is a good model for modern packet networks, such as Internet protocol (IP) and its adoption, general packet radio services (GPRS), into the wireless realm. The important elements are the network's packets, each of which either arrives at the destination or is lost due to congestion or corruption. Whenever there is at least one bit error in an arriving packet, the packet is considered lost and so discarded. A key property of the erasure channel is that the receiver knows which packets have been lost.

In the context of erasure channels, Albanese et al. [5] pioneered an unequal error protection scheme known as priority encoding transmission (PET). The PET scheme works with a family of channel codes, all of which have the same codeword length N, but different source lengths, $1 \le k \le N$. We consider only "perfect codes" which have the key property that the receipt of any k out of the N symbols in a codeword is sufficient to recover the k source symbols. The amount of redundancy $R_{N,k} = N/k$ determines the strength of the code, where smaller values of k correspond to stronger codes.

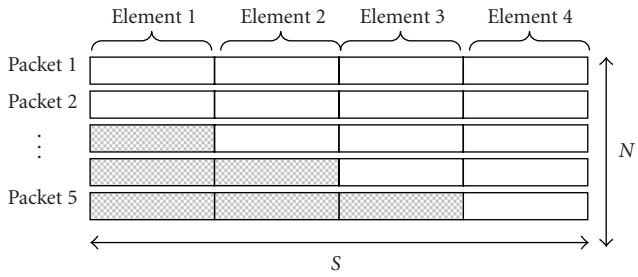


Figure 1: An example of PET arrangement of source elements into packets. Four elements are arranged into $N = 5$ packets with size S bytes. Elements $\mathscr{E}_1$ to $\mathscr{E}_4$ are assigned $k = \{2, 3, 4, 5\}$, respectively. The white areas correspond to the elements' content while the shaded areas contain parity information.

We define a scalable data source to consist of groups of symbols, each of which is referred to as the "source element" $\mathscr{E}_q$ having L_q symbols. Although in our experiment, each symbol corresponds to one byte, the source symbol is not restricted to a particular unit. Given a scalable data source consisting of source elements $\mathscr{E}_1, \mathscr{E}_2, \ldots, \mathscr{E}_Q$ having uncoded lengths $L_1, L_2, \ldots, L_Q$ and channel code redundancies $R_{N,k_1} \geq R_{N,k_2} \geq \cdots \geq R_{N,k_Q}$, the PET scheme packages the encoded elements into N network packets, where source symbols from each element $\mathscr{E}_q$ occupy k_q packets. This arrangement guarantees that the receipt of any k packets is sufficient to recover all elements $\mathscr{E}_q$ with $k_q \leq k$. The total encoded transmission length is $\sum_q L_q R_{N,k_q}$, which must be arranged into N packets, each having a packet size of S bytes. Figure 1 shows an example of arranging $Q = 4$ elements into $N = 5$ packets. Consider element $\mathscr{E}_2$, which is assigned a $(5, 3)$ code. Since $k_2 = 3$, three out of the five packets contain the source element's L_2 symbols. The remaining $N - k_2 = 2$ packets contain parity information. Hence, receiving any three packets guarantees recovery of element $\mathscr{E}_2$ and also $\mathscr{E}_1$.

Given the PET scheme and a scalable data source, several strategies have been proposed to find the optimal channel code allocation for each source element under the condition that the total encoded transmission length is no greater than a specified maximum transmission length $L_{\max} = NS$ [6, 7, 8, 9, 10, 11, 12]. The optimization objective is an expected utility U which must be an additive function of the source elements that are correctly received. That is,

$$U = U_0 + \sum_{q=1}^{Q} U_q P_{N,k_q}, \tag{1}$$

where U_0 is the amount of utility at the receiver when no source element is received and P_{N,k_q} is the probability of recovering element $\mathscr{E}_q$, which is assigned an (N, k_q) code. This probability equals the probability of receiving at least k_q out of N packets for $k_q > 0$. If a source element is not transmitted, we assign the otherwise meaningless value of $k_q = 0$ for which $R_{N,k_q} = 0$ and $P_{N,k_q} = 0$. As an example, for a scalable compressed image, $-U$ might represent the mean square error (MSE) of the reconstructed image, while U_q is the amount of reduction in MSE when element $\mathscr{E}_q$ is recovered correctly. In the event of losing all source elements, the reconstructed image is "blank" so $-U_0$ corresponds to the largest MSE and is equal to the variance of the original image. The term U_0 is included only for completeness; it plays no role in the intuitive or computational aspects of the optimization problem.

Unfortunately, these optimization strategies rely upon the PET encoding scheme. This requires all of the encoded source elements to be distributed across the same N packets. Given a small packet size and large amount of data, the encoder must use a family of perfect codes with large values of N. For instance, transmitting a 1 MB source using ATM cells with a packet size of 48 bytes requires $N = 21,000$. This imposes a huge computational burden on both the encoder and the decoder.

In this paper, we propose a strategy for optimally assigning code redundancies to source elements under two constraints. One constraint is transmission length, which limits the amount of encoded data being transmitted through the channel. The second constraint is the length of the channel codewords. The impact of this constraint depends on the channel packet size and the amount of data to be transmitted. In Sections 2 and 3, we explore the nature of scalable data and the erasure channel model. We coin the term "cluster of packets" (COP) to refer to a collection of network packets whose elements are jointly protected according to the PET arrangement illustrated in Figure 1. Section 4 reviews the code redundancy assignment strategy under the condition that all elements are arranged into a single COP; accordingly, we identify this as the "UniCOP assignment" strategy.

In Section 5, we outline the proposed strategy for assigning source elements into several COPs, each of which is made up of at most N channel packets, where N is the length of the channel codewords. Whereas packets are encoded jointly within any given COP, separate COPs are encoded independently. The need for multiple COPs arises when the maximum transmission length is larger than the specified COP size, NS. We use the term "MultiCOP assignment" when referring to this strategy. Given arrangement of source elements into COPs together with a maximum transmission length, we find the optimal code redundancy $R_{N,k}$ for each source element so as to maximize the expected utility U. Section 6 provides experimental results in the context of JPEG2000 data streams.

2. SCALABLE DATA

Scalable data is composed of nested elements. The compression of these elements generally imposes dependencies among the elements. This means that certain elements cannot be correctly decoded without first successfully decoding certain earlier elements. Figure 2 provides an example of dependency structure in a scalable source. Each "column" of elements $\mathscr{E}_{1,y}, \mathscr{E}_{2,y}, \ldots, \mathscr{E}_{X,y}$ has a simple chain of dependencies, which is expressed as $\mathscr{E}_{1,y} \prec \mathscr{E}_{2,y} \prec \cdots \prec \mathscr{E}_{X,y}$. This means that the element $\mathscr{E}_{1,y}$ must be recovered before the

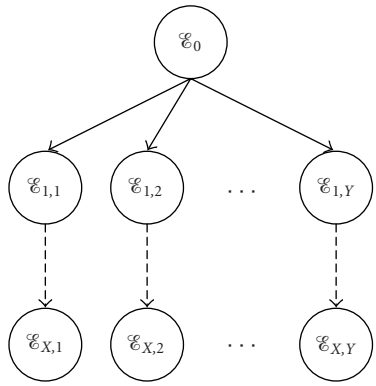


Figure 2: Example of dependency structure of scalable sources.

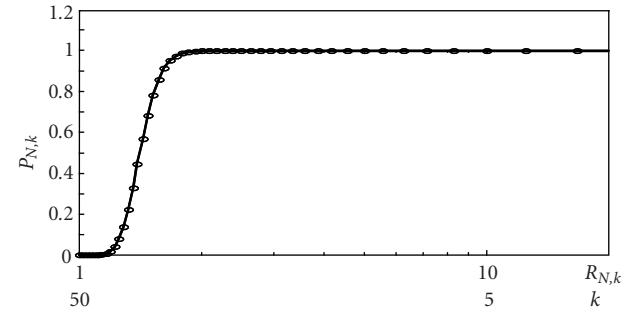


Figure 3: Example of $P_{N,k}$ versus $R_{N,k}$ characteristic with $N = 50$ and $p = 0.3$.

information in element $\mathscr{E}_{2,y}$ can be used and so forth. Since each column depends on element $\mathscr{E}_0$, this element must be recovered prior to the attempt to recover the first element of every column. There is, however, no dependency between the columns, that is, $\mathscr{E}_{x,y} \not\prec \mathscr{E}_{\bar{x},\bar{y}}$ and $\mathscr{E}_{x,y} \not\succ \mathscr{E}_{\bar{x},\bar{y}}$, $y \neq \bar{y}$. Hence, the elements from one column can be recovered without having to recover any elements belonging to other columns.

An image compressed with JPEG2000 serves as a good example, since it can have a combination of dependent and independent elements. Dependencies exist between successive "quality layers" within the JPEG2000 data stream, where an element which contributes to a higher quality layer cannot be decoded without first decoding elements from lower quality layers. JPEG2000 also contains elements which exhibit no such dependencies. In particular, subbands from different levels in the discrete wavelet transform (DWT) are coded and represented independently within the data stream. Similarly, separate colour channels within a colour image are also coded and represented independently within the data stream.

Elements of the JPEG2000 compressed data stream form a tree structure, as depicted in Figure 2. The data stream header becomes the "root" element. The "branches" correspond to independently coded precincts, each of which is decomposed into a set of elements with linear dependencies.

3. CHANNEL MODEL

The channel model we use is that of an erasure channel, having two important properties. One property is that packets are either received without any error or discarded due to corruption or congestion. Secondly, the receiver knows exactly which packets have been lost. We assume that the channel packet loss process is i.i.d., meaning that every packet has the same loss probability p and the loss of one packet does not influence the likelihood of losing other packets. To compare the effect of different packet sizes, it is useful to express the probability p in terms of a bit error probability or bit error rate (BER) ϵ. To this end, we will assume that packet loss arises from random bit errors in an underlying binary symmetric channel. The probability of losing any packet with size S bytes is then $p = 1 - (1 - \epsilon)^{8S}$. The probability of receiving at least k out of N packets with no error is then

$$P_{N,k} = \sum_{i=k}^{N} \binom{N}{i} (1-p)^i p^{N-i}. \tag{2}$$

Figure 3 shows an example of the relationship between $P_{N,k}$ and $R_{N,k}$ for the case $p = 0.3$. Evidently, $P_{N,k}$ is monotonically increasing with $R_{N,k}$. Significantly, however, the curve is not convex.

It is convenient to parametrize $P_{N,k}$ and $R_{N,k}$ by a single parameter

$$r = \begin{cases} N + 1 - k, & k > 0, \\ 0, & k = 0, \end{cases} \tag{3}$$

and to assume N implicitly for simpler notation so that

$$P(r) = P_{N,N+1-r}, \qquad R(r) = R_{N,N+1-r} \tag{4}$$

for $r = 1, \ldots, N$. It is also convenient to define

$$P(0) = R(0) = 0. \tag{5}$$

The parameter r is more intuitive than k since r increases in the same direction as $P(r)$ and $R(r)$. The special case $r = 0$ means that the relevant element is not transmitted at all.

4. UNICOP ASSIGNMENT

We review the problem of assigning an optimal set of channel codes to the elements of a scalable data source, subject to the assumption that all source elements will be packed into the same set of N channel packets, where N is the codeword length. The number of packets N and packet size S are fixed. This is the problem addressed in [6, 7, 8, 9, 10, 11, 12], which we identified earlier as the UniCOP assignment problem. Puri and Ramchandran [6] provided an optimization technique based on the method of Lagrange multipliers to find the channel code allocation. Mohr et al. [7] proposed a local search algorithm and later a faster algorithm [8] which is essentially a Lagrangian optimization. Stankovic et al. [11]

also presented a local search approach based on a fast iterative algorithm, which is faster than [8]. All these schemes assume that the source has convex utility-length characteristic. Stockhammer and Buchner [9] presented a dynamic programming approach which finds an optimal solution for convex utility-length characteristics. However, for general utility-length characteristics, the scheme is close to optimal. Dumitrescu et al. [10] proposed an approach based on a global search, which finds a globally optimal solution for both convex and nonconvex utility-length characteristics with similar computation complexity. However, for convex sources, the complexity is lower since it need not take into account the constraint from the PET framework that the amount of channel code redundancy must be nonincreasing. The UniCOP assignment strategy we discuss below is based on a Lagrangian optimization similar to [6]. However, this scheme not only works for sources with convex utility-length characteristic but also applies to general utility-length characteristics. Unlike [10], the complexity in both cases is about the same and the proposed scheme does not need to explicitly include the PET constraint since the solution will always satisfy that constraint. Most significantly, the UniCOP assignment strategy presented here serves as a stepping stone to the "MultiCOP assignment" in Section 5, where the behaviour with nonconvex sources will become important.

Suppose that the data source contains Q elements and each source element $\mathscr{E}_q$ has a fixed number of source symbols L_q. We assume that the data source has a simple chain of dependencies $\mathscr{E}_1 \prec \mathscr{E}_2 \prec \cdots \prec \mathscr{E}_Q$. This dependency will in fact impose a constraint that the code redundancy of the source elements must be nonincreasing, $R_{N,k_1} \geq R_{N,k_2} \geq \cdots \geq R_{N,k_Q}$, equivalently, $r_1 \geq r_2 \geq \cdots \geq r_Q$, such that the recovery of the element $\mathscr{E}_q$ guarantees the recovery of the elements $\mathscr{E}_1$ to $\mathscr{E}_{q-1}$. Generally, the utility-length characteristic of the data source can be either convex or nonconvex. To impart intuition, we begin by considering the former case in which the source utility-length characteristic is convex, as illustrated in Figure 4. That is,

$$\frac{U_1}{L_1} \geq \frac{U_2}{L_2} \geq \cdots \geq \frac{U_Q}{L_Q}. \tag{6}$$

We will later need to consider nonconvex utility-length characteristics when extending the protection assignment algorithm to multiple COPs even if the original source's utility-length characteristic was convex. Nevertheless, we will defer the generalization to nonconvex sources for the moment until Section 4.2 so as to provide a more accessible introduction to ideas.

4.1. Convex sources

To develop the algorithm for optimizing the overall utility U, we temporarily ignore the constraint $r_1 \geq \cdots \geq r_Q$, which arises from the dependence between source elements. We will show later that the solution we obtain will always satisfy this constraint by virtue of the source convexity. Our optimization problem is to optimize the utility function given in (1),

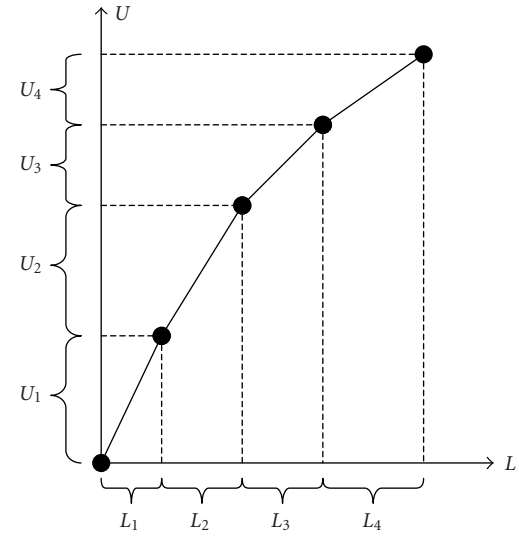


Figure 4: Example of convex utility-length characteristic for a scalable source consisting of four elements with a simple chain of dependencies.

subject to the overall transmission length constraint

$$L = \sum_{q=1}^{Q} L_q R(r_q) \leq L_{\max}. \tag{7}$$

This constrained optimization problem may be converted to a family of unconstrained optimization problems parametrized by a quantity $\lambda > 0$. Specifically, let $U^{(\lambda)}$ and $L^{(\lambda)}$ denote the expected utility and transmission length associated with the set $\{r_q^{(\lambda)}\}_{1\leq q\leq Q}$, which maximize the functional

$$\begin{aligned} J^{(\lambda)} &= U^{(\lambda)} - \lambda L^{(\lambda)} \\ &= \sum_{q=1}^{Q} U_q P(r_q^{(\lambda)}) - \lambda L_q R(r_q^{(\lambda)}). \end{aligned} \tag{8}$$

We omit the term U_0 since it only introduces an offset to the optimization expression and hence does not impact its solution. Evidently, it is impossible to increase U beyond $U^{(\lambda)}$ without also increasing L beyond $L^{(\lambda)}$. Thus if we can find λ such that $L^{(\lambda)} = L_{\max}$, the set $\{r_q^{(\lambda)}\}$ will form an optimal solution to our constrained problem. In practice, the discrete nature of the problem may prevent us from finding a value λ such that $L^{(\lambda)}$ is exactly equal to $L_{\max}$, but if the source elements are small enough, we should be justified in ignoring this small source of suboptimality and selecting the smallest value of λ such that $L^{(\lambda)} \leq L_{\max}$. The unconstrained optimization problem decomposes into a collection of Q separate maximization problems. In particular, we seek $r_q^{(\lambda)}$ which maximizes

$$J_q^{(\lambda)} = U_q P(r_q^{(\lambda)}) - \lambda L_q R(r_q^{(\lambda)}) \tag{9}$$

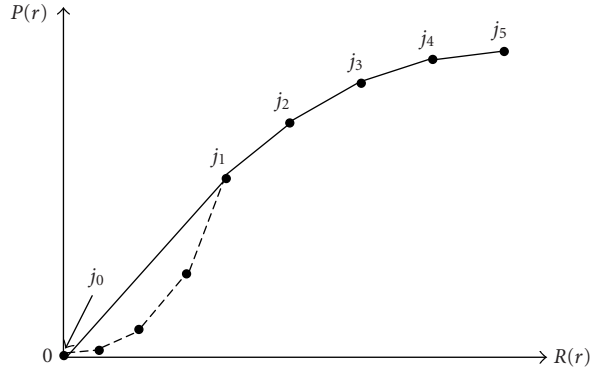


FIGURE 5: Elements of a convex hull set are the vertices $\{j_0, j_1, \ldots, j_5\}$ which lie on the convex hull of the $P(r)$ versus $R(r)$ characteristic.

for each $q = 1, 2, \ldots, Q$. Equivalently, $r_q^{(\lambda)}$ is the value of r that maximizes the expression

$$P(r) - \lambda_q R(r), \tag{10}$$

where $\lambda_q = \lambda L_q / U_q$. This optimization problem arises in other contexts, such as the optimal truncation of embedded compressed bitstreams [13, Section 8.2]. It is known that the solution $r_q^{(\lambda)}$ must be a member of the set $\mathcal{H}_C$ which describes the vertices of the convex hull of the $P(r)$ versus $R(r)$ characteristic [13, Section 8.2], as illustrated in Figure 5. Then, if $0 = j_0 < j_1 < \cdots < j_I = N$ is an enumeration of the elements in $\mathcal{H}_C$, and

$$S_C(i) = \begin{cases} \dfrac{P(j_i) - P(j_{i-1})}{R(j_i) - R(j_{i-1})}, & i > 0, \\ \infty, & i = 0, \end{cases} \tag{11}$$

are the "slope" values on the convex hull, then $S_C(0) \geq S_C(1) \geq \cdots \geq S_C(I)$. The solution to our optimization problem is obtained by finding the maximum value of $j_i \in \mathcal{H}_C$, which satisfies

$$P(j_i) - \lambda_q R(j_i) \geq P(j_{i-1}) - \lambda_q R(j_{i-1}). \tag{12}$$

Specifically,

$$\begin{aligned} r_q^{(\lambda)} &= \max \left\{ j_i \in \mathcal{H}_C \mid \frac{P(j_i) - P(j_{i-1})}{R(j_i) - R(j_{i-1})} \geq \lambda_q \right\} \\ &= \max \{ j_i \in \mathcal{H}_C \mid S_C(i) \geq \lambda_q \}. \end{aligned} \tag{13}$$

Given λ, the complexity of finding a set of optimal solutions $\{r_q^{(\lambda)}\}$ is $\mathbb{O}(IQ)$. Our algorithm first finds the largest λ such that $L^{(\lambda)} < L_{\max}$ and then employs a bisection search to find λ^{opt}, where $L^{(\lambda^{\text{opt}})} \lessapprox L_{\max}$. The number of iteration required to search for λ^{opt} is bounded by the computation precision, and the bisection search algorithm typically requires a small number of iterations to find λ^{opt}. In our experiments, the number of iterations is typically fewer than 15, which is usually much smaller than I or Q. It is also worth noting that the number of iterations required to find λ^{opt} is independent of other parameters, such as the number of source elements Q, the packet size S, and the codeword length N.

All that remains now is to show that this solution will always satisfy the necessary constraint $r_1 \geq r_2 \geq \cdots \geq r_Q$. To this end, observe that our source convexity assumption implies that $L_q/U_q \leq L_{q+1}/U_{q+1}$ so that

$$\{ j_i \in \mathcal{H}_C \mid S_C(i) \geq \lambda_q \} \supseteq \{ j_i \in \mathcal{H}_C \mid S_C(i) \geq \lambda_{q+1} \}. \tag{14}$$

It follows that

$$\begin{aligned} r_q^{(\lambda)} &= \max \{ j_i \in \mathcal{H}_C \mid S_C(i) \geq \lambda_q \} \\ &\geq \max \{ j_i \in \mathcal{H}_C \mid S_C(i) \geq \lambda_{q+1} \} = r_{q+1}^{(\lambda)}, \quad \forall q. \end{aligned} \tag{15}$$

4.2. Nonconvex sources

In the previous section, we restricted our attention to convex source utility-length characteristics, but did not impose any prior assumption on the convexity of the $P(r)$ versus $R(r)$ channel coding characteristic. As already seen in Figure 3, the $P(r)$ versus $R(r)$ characteristic is not generally convex. We found that the optimal solution is always drawn from the convex hull set $\mathcal{H}_C$ and that the optimization problem amounts to a trivial element-wise optimization problem in which $r_q^{(\lambda)}$ is assigned to the largest element $j_i \in \mathcal{H}_C$ whose slope $S_C(i)$ is no smaller than $\lambda L_q/U_q$.

In this section, we abandon our assumption on source convexity. We begin by showing that in this case, the optimal solution involves only those protection strengths r which belong to the convex hull $\mathcal{H}_C$ of the channel code's performance characteristic. We then show that the optimal protection assignment depends only on the convex hull of the source utility-length characteristic and that it may be found using the comparatively trivial methods previously described.

4.2.1. Sufficiency of the channel coding convex hull $\mathcal{H}_C$

Lemma 1. *Suppose that $\{r_q^{(\lambda)}\}_{1 \leq q \leq Q}$ is the collection of channel code indices which maximizes $J^{(\lambda)}$ subject to the ordering constraint $r_1^{(\lambda)} \geq r_2^{(\lambda)} \geq \cdots \geq r_Q^{(\lambda)}$. Then $r_q^{(\lambda)} \in \mathcal{H}_C$ for all q. More precisely, whenever there is $r_q \notin \mathcal{H}_C$ yielding $\bar{J}(\lambda)$, there is always another $r_q \in \mathcal{H}_C$, which yield $J^{(\lambda)} \geq \bar{J}(\lambda)$.*

Proof. As before, let $0 = j_0 < j_1 < \cdots < j_I$ be an enumeration of the elements in $\mathcal{H}_C$. For each $j_i \in \mathcal{H}_C$, define $\mathcal{F}_i = \{ r_q^{(\lambda)} \mid j_i < r_q^{(\lambda)} < j_{i+1} \}$. For convenience, we define $j_{I+1} = \infty$ so that the last of these sets $\mathcal{F}_I$ is well defined. The objective of the proof is to show that all of these sets $\mathcal{F}_i$ must be empty. To this end, suppose that some $\mathcal{F}_i$ is nonempty and let $\bar{r}_1 < \bar{r}_2 \cdots < \bar{r}_Z$ be an enumeration of its elements. For each $\bar{r}_z \in \mathcal{F}_i$, let $\bar{U}_z$ and $\bar{L}_z$ be the combined utilities and lengths of all source elements which were assigned $r_q^{(\lambda)} = \bar{r}_z$. That is,

$$\bar{U}_z = \sum_{q \ni r_q^{(\lambda)} = \bar{r}_z} U_q, \qquad \bar{L}_z = \sum_{q \ni r_q^{(\lambda)} = \bar{r}_z} L_q. \tag{16}$$

For each $z < Z$, we could assign the alternate value of $\bar{r}_{z+1}$ to all of the source elements with $r_q^{(\lambda)} = \bar{r}_z$ without violating the ordering constraint on $\bar{r}_q^{(\lambda)}$. This adjustment would result in a net increase in $J^{(\lambda)}$ of

$$\bar{U}_z[P(\bar{r}_{z+1}) - P(\bar{r}_z)] - \lambda\bar{L}_z[R(\bar{r}_{z+1}) - R(\bar{r}_z)]. \tag{17}$$

By hypothesis, we already have the optimal solution, so this alternative must be unfavourable, meaning that

$$\frac{P(\bar{r}_{z+1}) - P(\bar{r}_z)}{R(\bar{r}_{z+1}) - R(\bar{r}_z)} \le \lambda\frac{\bar{L}_z}{\bar{U}_z}. \tag{18}$$

Similarly, for any $z \le Z$, we could assign the alternate value of $\bar{r}_{z-1}$ to the same source elements (where we identify $\bar{r}_0$ with j_i for completeness) again without violating our ordering constraint. The fact that the present solution is optimal means that

$$\frac{P(\bar{r}_z) - P(\bar{r}_{z-1})}{R(\bar{r}_z) - R(\bar{r}_{z-1})} \ge \lambda\frac{\bar{L}_z}{\bar{U}_z}. \tag{19}$$

Proceeding by induction, we must have monotonically decreasing slopes

$$\begin{aligned}&\frac{P(\bar{r}_1) - P(j_i)}{R(\bar{r}_1) - R(j_i)}\\ &\quad\ge \frac{P(\bar{r}_2) - P(\bar{r}_1)}{R(\bar{r}_2) - R(\bar{r}_1)} \ge \cdots \ge \frac{P(\bar{r}_Z) - P(\bar{r}_{Z-1})}{R(\bar{r}_Z) - R(\bar{r}_{Z-1})}.\end{aligned} \tag{20}$$

It is convenient, for the moment, to ignore the pathological case $i = I$. Now since $\bar{r}_z \notin \mathcal{H}_C$, we must have

$$\begin{aligned}&\frac{P(j_{i+1}) - P(j_i)}{R(j_{i+1}) - R(j_i)}\\ &\quad\ge \frac{P(\bar{r}_1) - P(j_i)}{R(\bar{r}_1) - R(j_i)} \ge \cdots \ge \frac{P(\bar{r}_Z) - P(\bar{r}_{Z-1})}{R(\bar{r}_Z) - R(\bar{r}_{Z-1})},\end{aligned} \tag{21}$$

as illustrated in Figure 6. So, for any given $z \ge 1$, we must have

$$\frac{P(j_{i+1}) - P(\bar{r}_z)}{R(j_{i+1}) - R(\bar{r}_z)} \ge \frac{P(\bar{r}_z) - P(\bar{r}_{z-1})}{R(\bar{r}_z) - R(\bar{r}_{z-1})} \ge \lambda\frac{\bar{L}_z}{\bar{U}_z}, \tag{22}$$

meaning that all of the source elements which are currently assigned $r_q^{(\lambda)} = \bar{r}_z$ could be assigned $r_q^{(\lambda)} = j_{i+1}$ instead without decreasing the contribution of these source elements to $J^{(\lambda)}$. Doing this for all z simultaneously would not violate the ordering constraint, meaning that there is another solution, which is at least as good as the one claimed to be optimal, in which $\mathcal{F}_i$ is empty.

For the case $i = I$, the fact that $\bar{r}_1 \notin \mathcal{H}_C$ and that there are no larger values of r which belong to the convex hull means that $(P(\bar{r}_1) - P(j_i))/(R(\bar{r}_1) - R(j_i)) \le 0$ and hence $(P(\bar{r}_z) - P(\bar{r}_{z-1}))/(R(\bar{r}_z) - R(\bar{r}_{z-1})) \le 0$ for each z. But this contradicts (19) since $\lambda(\bar{L}_z/\bar{U}_z)$ is strictly positive. Therefore, $\mathcal{F}_I$ is also empty. □

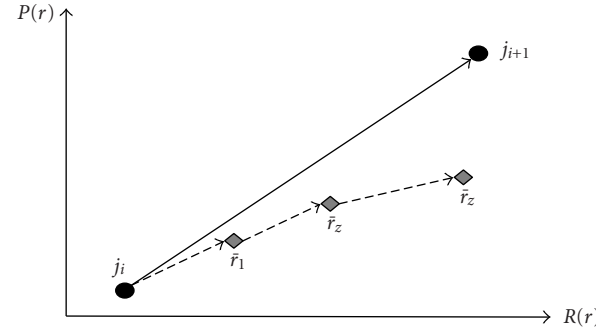


FIGURE 6: The parameters $\bar{r}_1, \ldots, \bar{r}_Z$ between j_i and j_{i+1} are not part of convex hull points and have decreasing slopes.

4.2.2. *Sufficiency of the source convex hull* $\mathcal{H}_S$

In the previous section, we showed that we may restrict our attention to channel codes belonging to the convex hull set, that is, $r \in \mathcal{H}_C$, regardless of the source convexity. In this section, we show that we may also restrict our attention to the convex hull of the source utility-length characteristic.

Since the solution to our optimization problem satisfies $r_1^{(\lambda)} \ge r_2^{(\lambda)} \ge \cdots \ge r_Q^{(\lambda)}$, it may equivalently be described in terms of a collection of thresholds $1 \le t_i^{(\lambda)} \le Q$ which we define according to

$$t_i^{(\lambda)} = \max\{q \mid r_q^{(\lambda)} \ge j_i\}, \tag{23}$$

where $0 = j_0 < j_1 < \cdots < j_I = N$ is our enumeration of $\mathcal{H}_C$. For example, consider a source with $Q = 6$ elements and a channel code convex hull $\mathcal{H}_C$ with $j_i \in \{0, 1, 2, \ldots, 6\}$. Suppose that these elements are assigned

$$(r_1, r_2, \ldots, r_6) = (5, 3, 2, 1, 1, 0). \tag{24}$$

Then, elements that are assigned at least $j_0 = 0$ correspond to all the six r's and so $t_0 = 6$. Similarly, elements that are assigned at least $j_1 = 1$ correspond to the first five r's and so, $t_1 = 5$. Performing the same computation as above for the remaining j_i produces

$$(t_0, t_1, \ldots, t_6) = (6, 5, 3, 2, 1, 1, 0). \tag{25}$$

Evidently, the thresholds are ordered according to $Q = t_0^{(\lambda)} \ge t_1^{(\lambda)} \ge \cdots \ge t_I^{(\lambda)}$. The $r_q^{(\lambda)}$ values may be recovered from this threshold description according to

$$r_q^{(\lambda)} = \max\{j_i \in \mathcal{H}_C \mid t_i^{(\lambda)} \ge q\}. \tag{26}$$

Using the same example above, given the channel code convex hull points $\{0, 1, 2, \ldots, 6\}$ and a set of thresholds (25), possible threshold values for $\mathcal{C}_1$ are $(t_0, t_1, \ldots, t_5)$ and so, $r_1 = 5$. Similarly, possible threshold values for $\mathcal{C}_2$ are $(t_0, \ldots, t_3)$ and so, $r_2 = 3$. Performing the same computation as above for the remaining elements will produce the original code (24). Now, the unconstrained optimization problem from

(8) may be expressed as

$$\begin{aligned} J^{(\lambda)} &= \sum_{q=1}^{t_1^{(\lambda)}} U_q P(j_1) - \lambda L_q R(j_1) \\ &\quad + \sum_{q=1}^{t_2^{(\lambda)}} U_q [P(j_2) - P(j_1)] - \lambda L_q [R(j_2) - R(j_1)] + \cdots \\ &= \sum_{i=1}^{I} \underbrace{\left[\sum_{q=1}^{t_i^{(\lambda)}} U_q \dot{P}_i - \lambda L_q \dot{R}_i \right]}_{O_i^{(\lambda)}}, \end{aligned} \tag{27}$$

where

$$\dot{P}_i \triangleq P(j_i) - P(j_{i-1}), \qquad \dot{R}_i \triangleq R(j_i) - R(j_{i-1}). \tag{28}$$

If we temporarily ignore the constraint that the thresholds must be properly ordered according to $t_1^{(\lambda)} \geq t_2^{(\lambda)} \geq \cdots \geq t_I^{(\lambda)}$, we may maximize $J^{(\lambda)}$ by maximizing each of the terms $O_i^{(\lambda)}$ separately. We will find that we are justified in doing this since the solution will always satisfy the threshold ordering constraint. Maximizing $O_i^{(\lambda)}$ is equivalent to finding $t_i^{(\lambda)}$, which maximize

$$\sum_{q=1}^{t_i^{(\lambda)}} U_q - \dot{\lambda}_i L_q, \tag{29}$$

where $\dot{\lambda}_i = \lambda \dot{R}_i / \dot{P}_i$. The same problem arises in connection with optimal truncation of embedded source codes[1] [13, Section 8.2]. It is known that the solutions $t_i^{(\lambda)}$ must be drawn from the convex hull set $\mathcal{H}_S$. Similar to $\mathcal{H}_C$, $\mathcal{H}_S$ contains vertices lying on the convex hull curve of the utility-length characteristic. Let $0 = h_0 < h_1 < \cdots < h_H = Q$ be an enumeration of the elements of $\mathcal{H}_S$ and let

$$S_S(n) = \begin{cases} \dfrac{\sum_{q=h_{n-1}+1}^{h_n} Uq}{\sum_{q=h_{n-1}+1}^{h_n} Lq}, & n > 0, \\ \infty, & n = 0, \end{cases} \tag{30}$$

be the monotonically decreasing slopes associated with $\mathcal{H}_S$. Then

$$\begin{aligned} t_i^{(\lambda)} &= \max \left\{ h_n \in \mathcal{H}_S \mid \sum_{q=h_{n-1}+1}^{h_n} U_q - \dot{\lambda}_i L_q \geq 0 \right\} \\ &= \max \left\{ h_n \in \mathcal{H}_S \mid \frac{\sum_{q=h_{n-1}+1}^{h_n} U_q}{\sum_{q=h_{n-1}+1}^{h_n} L_q} \geq \dot{\lambda}_i \right\} \\ &= \max \{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \dot{\lambda}_i \} \\ &= \max \{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \lambda \dot{R}_i / \dot{P}_i \}. \end{aligned} \tag{31}$$

[1] In fact, this is the same problem as in Section 4.1 except that $P(r)$ and $R(r)$ are replaced with $\sum_{q=1}^{t} U_q$ and $\sum_{q=1}^{t} L_q$.

Finally, observe that

$$\frac{\dot{R}_i}{\dot{P}_i} = \frac{R(j_i) - R(j_{i-1})}{P(j_i) - P(j_{i-1})} = \frac{1}{S_C(i)}. \tag{32}$$

Monotonicity of the channel coding slopes $S_C(i)$ implies that $S_C(i) \geq S_C(i+1)$ and hence $\lambda / S_C(i) \leq \lambda / S_C(i+1)$. Then,

$$\begin{aligned} &\{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \lambda / S_C(i) \} \\ &\quad \supseteq \{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \lambda / S_C(i+1) \}. \end{aligned} \tag{33}$$

It follows that

$$\begin{aligned} t_i^{(\lambda)} &= \max \{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \lambda / S_C(i) \} \\ &\geq \max \{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \lambda / S_C(i+1) \} = t_{i+1}^{(\lambda)}. \end{aligned} \tag{34}$$

Therefore, the required ordering property $t_1^{(\lambda)} \geq t_2^{(\lambda)} \geq \cdots \geq t_I^{(\lambda)}$ is satisfied.

In summary, for each $j_i \in \mathcal{H}_C$, we find the threshold $t_i^{(\lambda)}$ from

$$t_i^{(\lambda)} = \max \{ h_n \in \mathcal{H}_S \mid S_S(n) \geq \lambda / S_C(i) \} \tag{35}$$

and then assign (26). The solution is guaranteed to be at least as good as any other channel code assignment, in the sense of maximizing $J^{(\lambda)}$ subject to $r_1^{(\lambda)} \geq r_2^{(\lambda)} \geq \cdots \geq r_Q^{(\lambda)}$, regardless of the convexity of the source or channel codes. The computational complexity is now $\mathbb{O}(IH)$ for each λ. Similar to the convex sources case, we employ the bisection search algorithm to find λ^{opt}.

5. MULTICOP ASSIGNMENT

In the UniCOP assignment strategy, we assume that either the packet size S or the codeword length N can be set sufficiently large so that the data source can always fit into N packets. Specifically, the UniCOP assignment holds under the following condition:

$$\sum_{q=1}^{Q} L_q R(r_q) \leq NS. \tag{36}$$

Recall from Figure 1 that NS is the COP size.

The choice of the packet size depends on the type of channel that the data is transmitted through. Some channels might have low BERs allowing the use of large packet sizes with a reasonably high probability of receiving error-free packets. However, wireless channels typically require small packets due to their much higher BER. Packaging a large amount of source data into small packets requires a large number of packets and hence long codewords. This is undesirable since it imposes a computational burden on both the channel encoder and, especially, the channel decoder.

If the entire collection of protected source elements cannot fit into a set of N packets of length S, more than one COP must be employed. When elements are arranged into COPs, we no longer have any guarantee that a source element

with a stronger code can be recovered whenever a source element with a weaker code is recovered. The code redundancy assignment strategy described in Section 4 relies upon this property in order to ensure that element dependencies are satisfied, allowing us to use (1) for the expected utility.

5.1. Code redundancy optimization

Consider a collection of C COPs $\{\mathscr{C}_1,\ldots,\mathscr{C}_C\}$ characterized by $\{(s_1,f_1),\ldots,(s_C,f_C)\}$, where s_c and f_c represent the indices of the first and the last source elements residing in the COP $\mathscr{C}_c$. We assume that the source elements have a simple chain of dependencies $\mathscr{E}_1 \prec \mathscr{E}_2 \prec \cdots \prec \mathscr{E}_Q$ such that prior to recovering an element $\mathscr{E}_q$, all preceding elements $\mathscr{E}_1,\ldots,\mathscr{E}_{q-1}$ must be recovered first. Within each COP $\mathscr{C}_i$, we can still constrain the code redundancies to satisfy

$$r_{s_i} \geq r_{s_i+1} \geq \cdots \geq r_{f_i} \tag{37}$$

and guarantee that no element in COP $\mathscr{C}_i$ will be recovered unless all of its dependencies within the same COP are also recovered. The probability $P(r_{f_i})$ of recovering the last element $\mathscr{E}_{f_i}$ thus denotes the probability that all elements in COP $\mathscr{C}_i$ are recovered successfully. Therefore, any element $\mathscr{E}_q$ in COP $\mathscr{C}_c$, which is correctly recovered from the channel, will be usable if and only if the last element of each earlier COP is recovered. This changes the expected utility in (1) to

$$U = U_0 + \sum_{c=1}^{C}\sum_{q=s_c}^{f_c}\left(U_q P(r_q)\prod_{i=1}^{c-1}P(r_{f_i})\right). \tag{38}$$

Our objective is to maximize this expression for U subject to the same total length constraint $L_{\max}$, as given in (7), and subject also to the constraint that

$$r_{s_c} \geq r_{s_c+1} \geq \cdots \geq r_{f_c} \tag{39}$$

for each COP $\mathscr{C}_c$. Similar to the UniCOP assignment strategy, this constrained optimization problem can be converted into a set of unconstrained optimization problems parametrized by λ. Specifically, we search for the smallest λ such that $L^{(\lambda)} \leq L_{\max}$, where $L^{(\lambda)}$ is the overall transmission length associated with the set $\{r_q^{(\lambda)}\}_{1\leq q\leq Q}$, which maximizes

$$\begin{aligned} J^{(\lambda)} &= U^{(\lambda)} - \lambda L^{(\lambda)} \\ &= \sum_{c=1}^{C}\sum_{q=s_c}^{f_c} U_q P(r_q^{(\lambda)})\prod_{i=1}^{c-1}P(r_{f_i}^{(\lambda)}) - \lambda L_q R(r_q^{(\lambda)}) \end{aligned} \tag{40}$$

subject to the constraint $r_{s_c} \geq r_{s_c+1} \geq \cdots \geq r_{f_c}$ for all c. This new functional turns out to be more difficult to optimize than that in (8) since the product terms in $U^{(\lambda)}$ couple the impact of code redundancy assignments for different elements. In fact, the optimization objective is generally multimodal exhibiting multiple local optima.

Nevertheless, it is possible to devise a simple optimization strategy, which rapidly converges to a local optimum, with good results in practice. Specifically, given an initial set of $\{r_q\}_{1\leq q\leq Q}$ and considering only one COP, $\mathscr{C}_c$, at a time, we can find a set of code redundancies $\{r_{s_c},\ldots,r_{f_c}\}$ which maximizes $J^{(\lambda)}$ subject to all other r_q's being held constant. The solution is sensitive to the initial $\{r_q\}$ set since the optimization problem is multimodal. However, as we shall see shortly in Section 5.2, since we build multiple COPs out of one COP, it is reasonable to set the initial values of $\{r_q\}$ equal to those obtained from the UniCOP assignment of Section 4. The UniCOP assignment works under the assumption that all encoded source elements can fit into one COP. This algorithm is guaranteed to converge as we cycle through each COP in turn, since the code redundancies for each COP either increase $J^{(\lambda)}$ or leave it unchanged, and the optimization objective is clearly bounded above by $\sum_q U_q$. The optimal solution for each COP is found by employing the scheme developed in Section 4. Our optimization objective for each COP $\mathscr{C}_c$ is to maximize a quantity

$$\begin{aligned} J_c^{(\lambda)} = \sum_{q=s_c}^{f_c} U_q\left[\prod_{i=1}^{c-1}P(r_{f_i}^{(\lambda)})\right]P(r_q^{(\lambda)}) \\ - \lambda L_q R(r_q^{(\lambda)}) + P(r_{f_c}^{(\lambda)})\Gamma_c \end{aligned} \tag{41}$$

while keeping code redundancies in other COPs constant. The last element $\mathscr{E}_{f_c}$ in COP $\mathscr{C}_c$ is unique since its recovery probability appears in the utility term of succeeding elements $\mathscr{E}_{f_c+1},\ldots,\mathscr{E}_Q$ which reside in COPs $\mathscr{C}_{c+1},\ldots,\mathscr{C}_C$. This effect is captured by the term

$$\Gamma_c = \sum_{m=c+1}^{C}\sum_{n=s_m}^{f_m} U_n P(r_n^{(\lambda)})\prod_{i=1,\, i\neq c}^{m-1}P(r_{f_i}^{(\lambda)}); \tag{42}$$

Γ_c can be considered as an additional contribution to the effective utility of $\mathscr{E}_{f_c}$.

Evidently, Γ_c is nonnegative, so it will always increase the effective utility of the last element in any COP $\mathscr{C}_c$, $c < C$. Even if the original source elements have a convex utility-length characteristic such that

$$\frac{U_{s_c}}{L_{s_c}} \geq \frac{U_{s_c+1}}{L_{s_c+1}} \geq \cdots \geq \frac{U_{f_c}}{L_{f_c}}, \tag{43}$$

the optimization of $J_c^{(\lambda)}$ subject to $r_{s_c} \geq r_{s_{c+1}} \geq \cdots \geq r_{f_c}$ involves the effective utilities

$$U_q' = \begin{cases} U_q\displaystyle\prod_{i=1}^{c-1}P(r_{f_i}^{(\lambda)}), & q = s_c,\ldots,f_c-1, \\ U_q\displaystyle\prod_{i=1}^{c-1}P(r_{f_i}^{(\lambda)}) + \Gamma_c, & q = f_c. \end{cases} \tag{44}$$

Apart from the last element $q = f_c$, U_q' is a scaled version of U_q involving the same scaling factor $\prod_{i=1}^{c-1}P(r_i^{(\lambda)})$ for each q. However, the last element $\mathscr{E}_{f_c}$ has an additional utility Γ_c which can destroy the convexity of the source effective utility-length characteristic. This phenomenon forms the principle motivation for the development in Section 4 of code redundancy assignment strategy, which is free from the assumption of convexity on the source or channel code characteristic.

In summary, the code redundancy assignment strategy for multiple COPs involves cycling through the COPs one at a time, holding the code redundancy for all COPs constant, and finding the values of $r_q^{(\lambda)}$, $s_c \le q \le f_c$, which maximize

$$J_c^{(\lambda)} = \sum_{q=s_c}^{f_c} U_q' P(r_q^{(\lambda)}) - \lambda L_q R(r_q^{(\lambda)}) \tag{45}$$

subject to the constraint $r_{s_c} \ge \cdots \ge r_{f_c}$. Maximization of $J_c^{(\lambda)}$ subject to $r_{s_c} \ge \cdots \ge r_{f_c}$, is achieved by using the strategy developed in Section 4, replacing each element's utility U_q with its current effective utility U_q'. Specifically, for each COP $\mathscr{C}_c$, we find a set of $\{t_i^{(\lambda)}\}$ which must be drawn from the convex hull set $\mathscr{H}_S^{(c)}$ of the source effective utility-length characteristic. Since U_{f_c}' is affected by $\{r_{f_c+1}, \ldots, r_Q\}$, elements in $\mathscr{H}_S^{(c)}$ may vary depending on these code redundancies and thus must be recomputed at each iteration of the algorithm. Then,

$$t_i^{(\lambda)} = \max\{h_n \in \mathscr{H}_S^{(c)} \mid S_S(n) \ge \lambda / S_C(i)\}, \tag{46}$$

where

$$S_S(n) = \begin{cases} \dfrac{\sum_{q=h_{n-1}+1}^{h_n} U_q'}{\sum_{q=h_{n-1}+1}^{h_n} L_q}, & n > 0 \\ \infty, & n = 0. \end{cases} \tag{47}$$

The solution $r_q^{(\lambda)}$ may be recovered from $t_i^{(\lambda)}$ using (26). As in the UniCOP case, we find the smallest value of λ such that the resulting solution satisfies $L^{(\lambda)} \le L_{\max}$. Similar to the UniCOP assignment of nonconvex sources, for each COP $\mathscr{C}_c$, the computation complexity is $\mathbb{O}(IH_c)$, where H_c is the number of elements in $\mathscr{H}_S^{(c)}$. Hence, in each iteration, it requires $\mathbb{O}(IH)$ computations, where $H = \sum_{c=1}^{C} H_c$. For some $\lambda > 0$, it typically requires fewer than 10 iterations for the solution to converge.

5.2. COP allocation algorithm

We are still left with the problem of determining the best allocation of elements to COPs subject to the constraint that the encoded source elements in any given COP should be no larger than NS. When $L_{\max}$ is larger than NS, the need to use multiple COPs is inevitable. The proposed algorithm starts by allocating all source elements to a single COP $\mathscr{C}_1$. Code redundancies are found by applying the UniCOP assignment strategy of Section 4. COP $\mathscr{C}_1$ is then split into two parts, the first of which contains as many elements as possible (f_1 as large as possible) while still having an encoded length $L_{\mathscr{C}_1}$ no larger than NS. At this point, the number of COPs is $C = 2$ and $\mathscr{C}_2$ does not generally satisfy $L_{\mathscr{C}_2} \le NS$.

The algorithm proceeds in an iterative sequence of steps. At the start of the tth step, there are C_t COPs, all but the last of which have encoded lengths no larger than NS. In this step, we first apply the MultiCOP code redundancy assignment algorithm of Section 5.1 to find a new set of $\{r_{s_c}, \ldots, r_{f_c}\}$ for each COP $\mathscr{C}_c$ maximizing the total expected utility subject to the overall length constraint $L_{\max}$. The new code redundancies produced by the MultiCOP assignment algorithm may cause one or more of the initial $C_t - 1$ COPs to violate the encoded length constraint of $L_{\mathscr{C}_c} \le NS$. In fact, as the algorithm proceeds, the encoded lengths of source elements assigned to all but the last COP tend to increase rather than decrease, as we shall argue later. The step is completed in one of two ways depending on whether or not this happens.

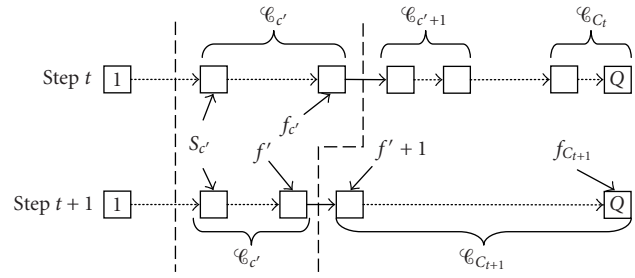


FIGURE 7: Case 1 of the COP allocation algorithm. At step t, $L_{\mathscr{C}_c}$ exceeds NS and hence is truncated. Its trailing elements and the rest of source elements are allocated to one COP, $\mathscr{C}_{C_{t+1}}$.

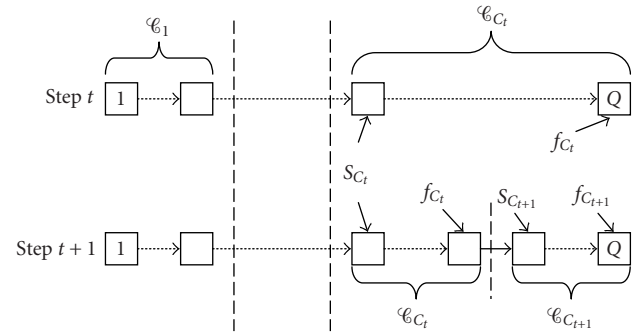


FIGURE 8: Case 2 of the COP allocation algorithm. At step t, the last COP is divided into two, the first of which, $\mathscr{C}_{C_t}$, satisfies NS.

Case 1 ($L_{\mathscr{C}_c} > NS$ for some $c < C_t$). Let $\mathscr{C}_{c'}$ be the first COP for which $L_{\mathscr{C}_{c'}} > NS$. In this case, we find the largest value of $f' \ge s_{c'}$ such that $\sum_{q=s_{c'}}^{f'} L_q R(r_q) \le NS$. COP $\mathscr{C}_{c'}$ is truncated by setting $f_{c'} = f'$ and all of the remaining source elements $\mathscr{E}_{f'+1}, \mathscr{E}_{f'+2}, \ldots, \mathscr{E}_Q$ are allocated to $\mathscr{C}_{c'+1}$. The algorithm proceeds in the next step with only $C_{t+1} = c' + 1 \le C_t$ COPs, all but the last of which satisfy the length constraint. Figure 7 illustrates this case.

Case 2 ($L_{C_c} \le NS$, for all $c < C_t$). In this case, we find the largest value of $f \ge s_{C_t}$ in order to satisfy $\sum_{q=s_{C_t}}^{f} L_q R(r_q) \le NS$, setting $f_{C_t} = f$. If $f = Q$, all source elements are already allocated to COPs, satisfying the length constraint, and their code redundancies are already jointly optimized, so we are done. Otherwise, the algorithm proceeds in the next step with $C_{t+1} = C_t + 1$ COPs, where $\mathscr{C}_{C_{t+1}}$ contains all of the remaining source elements $\mathscr{E}_{f+1}, \mathscr{E}_{f+2}, \ldots, \mathscr{E}_Q$. Figure 8 demonstrates this case.

To show that the algorithm must complete after a finite number of steps, observe first that the number of COPs must be bounded above by some quantity $M \leq Q$. Next, define an integer-valued functional

$$Z_t = \sum_{c=1}^{C_t} ||\mathscr{C}_c^{(t)}|| Q^{(M-c)}, \tag{48}$$

where $||\mathscr{C}_c^{(t)}||$ denotes the number of source elements allocated to COP $\mathscr{C}_c$ at the beginning of step t. This functional has the important property that each step in the allocation algorithm decreases Z_t. Since Z_t is always a positive finite integer, the algorithm must therefore complete in a finite number of steps. To see that each step does indeed decrease Z_t, consider the two cases. If step t falls into Case 1, with $\mathscr{C}_{c'}$ the COP whose contents are reduced, we have

$$\begin{aligned} Z_{t+1} &= \sum_{c=1}^{c'-1} ||\mathscr{C}_c^{(t)}|| Q^{(M-c)} + (f+1-s_{c'}^{(t)}) Q^{(M-c')} \\ &\quad + (Q-f) Q^{(M-c'-1)} \\ &= \sum_{c=1}^{c'} ||\mathscr{C}_c^{(t)}|| Q^{(M-c)} + (Q-f_{c'}^{(t)}) Q^{(M-c'-1)} \\ &\quad - (f_{c'}^{(t)} - f)\left(Q^{(M-c')} - Q^{(M-c'-1)}\right) \\ &< \sum_{c=1}^{C_t} ||\mathscr{C}_c^{(t)}|| Q^{(M-c)} + (Q-2)\cdot Q^{M-c'-1} \\ &\quad - \left(Q^{(M-c')} - Q^{(M-c'-1)}\right) \\ &< \sum_{c=1}^{C_t} ||\mathscr{C}_c^{(t)}|| Q^{(M-c)} = Z_t. \end{aligned} \tag{49}$$

If step t falls into Case 2, some of the source elements are moved from $\mathscr{C}_{C_t}$ to $\mathscr{C}_{C_t+1}$, where their contribution to Z_t is reduced by a factor of Q, so $Z_{t+1} < Z_t$.

The key property of our proposed COP allocation algorithm, which ensures its convergence, is that whenever a step does not split the final COP, it necessarily decreases the number of source elements contained in a previous COP $\mathscr{C}_{c'}$. The algorithm contains no provision for subsequently reconsidering this decision and moving some or all of these elements back into $\mathscr{C}_{c'}$. We claim that there is no need to revisit the decision to move elements out of $\mathscr{C}_{c'}$ for the following reason. Assuming that we do not alter the contents of any previous COPs (otherwise, the algorithm essentially restarts from that earlier COP boundary), by the time the allocation is completed, all source elements following the last element in $\mathscr{C}_{c'}$ should be allocated to COPs $\mathscr{C}_c$, with indices c at least as large as they were in step t. Considering (44), the effective utilities of these source elements will tend to be reduced relative to the effective utilities of the source elements allocated to $\mathscr{C}_1$ through $\mathscr{C}_{c'}$. Accordingly, one should expect the source elements allocated to $\mathscr{C}_1$ through $\mathscr{C}_{c'}$ to receive a larger share of the overall length budget $L_{\max}$, meaning that their coded lengths should be at least as large as they were in step t. While this is not a rigorous proof of optimality, it provides a strong justification for the proposed allocation algorithm. In practice, as the algorithm proceeds, we always observe that the code redundancies assigned to source elements in earlier COPs either remain unchanged or else increase.

5.3. Remarks on the effect of packet size

Throughout this paper, we have assumed that $P(r_q)$, the probability of receiving sufficient packets to decode an element $\mathscr{E}_q$, depends only on the selection of $r_q = N - k_q + 1$, where an (N, k_q) channel code is used. The value of N is fixed, but as discussed in Section 3, the value of $P(r_q)$ also depends upon the actual size of each encoded packet. We have taken this to be S, but our code redundancy assignment and COP allocation algorithms use S only as an upper bound for the packet size. If the maximum NS bytes are not used by any COP, each packet may actually be smaller than S. This, in turn, may alter the values of $P(r_q)$ so that our assigned codes are no longer optimal.

Fortunately, if the individual source elements are sufficiently small, the actual size of each COP should be approximately equal to its maximum value of NS, meaning that the actual packet size should be close to its maximum value of S. It is true that allocating more COPs, each with a smaller packet size, can yield higher expected utilities. However, rather than explicitly accounting for the effect of actual packet sizes within our optimization algorithm, various values for S are considered in an "outer optimization loop." In particular, for each value of S, we compute the channel coding characteristic described by $P(r_q)$ and $R(r_q)$ and then invoke our COP allocation and code redundancy optimization algorithm. Section 6 presents expected utility results obtained for various values of S. One potential limitation of this strategy is that the packet size S is essentially being forced to take on the same value within each COP. We have not considered the possibility of allowing different packet sizes or even different channel code lengths N for each COP.

6. COMPARATIVE RESULTS

In this section, we compare the total expected utility of a compressed image at the destination whose code redundancies have been determined using the UniCOP and MultiCOP assignment strategies described in Sections 4 and 5. We select a code length $N = 100$, a maximum transmission length $L_{\max} = 1\,000\,000$ bytes, a range of BER ϵ, and packet sizes S. The scalable data source used in these experiments is a 2560×2048 JPEG2000 compressed image, decomposed into 6 resolution levels. The image is grayscale exhibiting only one colour component and we treat the entire image as one tile component. Each resolution level is divided into a collection of precincts with size 128×128 samples, resulting in a total of 429 precincts. Each precinct is further decomposed into 12 quality elements. Overall, there are 5149 elements, treating each quality element and the data stream header as a source element. It is necessary to create a large number of source elements so as to minimize the impact of the discrete nature of our optimization problem, which may otherwise produce suboptimal solutions as discussed in Section 4.1.

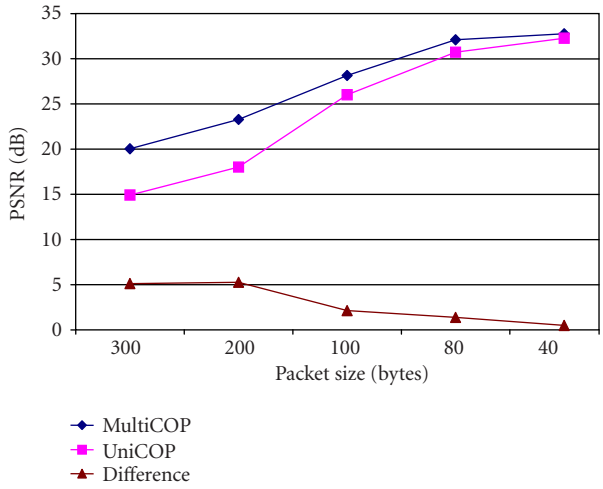


Figure 9: Comparative performance between UniCOP and MultiCOP assignment strategies with $\epsilon = 10^{-3}$.

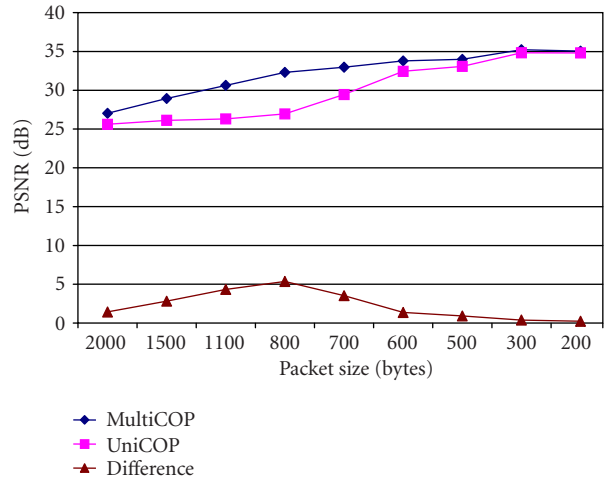


Figure 10: Comparative performance between UniCOP and MultiCOP assignment strategies with $\epsilon = 10^{-4}$.

We arrange the source elements into a linear sequence exhibiting a simple chain of dependencies with a convex utility-length characteristic. For simplicity, we assume that in the event, where any part of any element is corrupted, the entire element will be rendered useless along with all subsequent elements which depend upon it.[2] The UniCOP results were obtained by using the UniCOP assignment under the assumption that all source elements can be arranged into one COP. The encoded elements are then assigned to multiple COPs, whenever this is demanded by the constraint *NS*. The utility measure used here is a negated MSE, and the total expected utility is conveniently expressed in terms of peak signal-to-noise ratio (PSNR).[3] An improvement in the expected utility is equivalent to an increase in PSNR. To get a reasonable approximation of the total expected utility for each value of the packet size parameter *S*, the number of experiments which we run to find the overall expected utility is adjusted according to the packet loss probability. The MultiCOP results were based on the MultiCOP assignment algorithm, which progressively allocates source elements to COPs and assigns code redundancies to source elements accordingly. Figures 9, 10, and 11 compare the UniCOP results with those obtained by using the MultiCOP assignment strategy.

If all of the coded source elements are able to fit inside a single COP subject to the constraints determined by *N* and *S*, the UniCOP assignment will be optimal. Moreover, in this case, the UniCOP and MultiCOP strategies produce identical solutions. Otherwise, source elements must be assigned to multiple COPs, violating some of the assumptions underlying the UniCOP assignment strategy. In particular, the

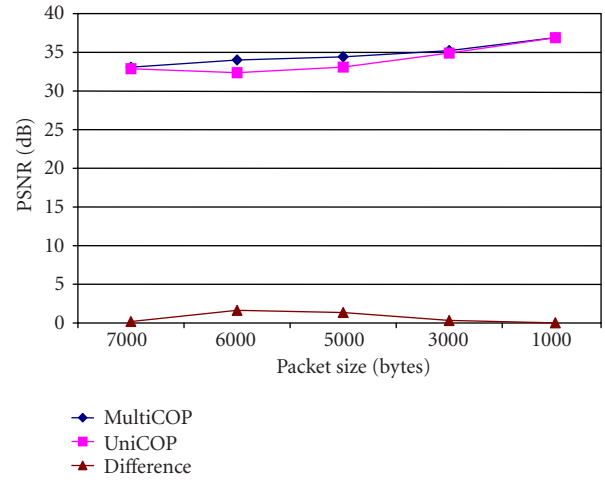


Figure 11: Comparative performance between UniCOP and MultiCOP assignment strategies with $\epsilon = 10^{-5}$.

recovery of any element $\mathscr{E}_q$ no longer guarantees the recovery of all the preceding elements $\mathscr{E}_1, \ldots, \mathscr{E}_{q-1}$ whose code redundancies are at least equal to that of $\mathscr{E}_q$. In this case, we would expect to see the MultiCOP assignment strategy providing superior performance. Figures 9, 10, and 11 show that both UniCOP and MultiCOP assignment strategies produce higher PSNR when the packet sizes are small. This is due to the fact that for a given BER, the packet loss probability decreases with decreasing the packet size. Low packet loss probability allows the elements to be assigned with weak codes and hence to be encoded with lower amount of redundancy. Therefore, it is possible to transmit more encoded elements without exceeding the maximum length $L_{\max}$.

The PSNR values from the MultiCOP assignment strategy are always higher relative to the UniCOP case for a given packet size and BER. The MultiCOP assignment process

[2] In practice, this assumption is excessively conservative since JPEG2000 decoders are able to recover well from some types of error.

[3] Peak signal-to-noise ratio is defined as $10\log(P^2/\text{MSE})$, where P is the peak-to-peak signal amplitude. In this case, $P = 255$ since we are working with 8-bit images.

described in Section 5 increases the expected utilities of the elements in the earlier COPs relative to those of the elements in the later COPs. This causes the same or a stronger code to be assigned to the elements in the earlier COPs. As a result, the elements in the earlier COPs are not corrupted as easily by packet loss.

The improvement in the PSNR for the MultiCOP assignment also depends on the BER ϵ. At high BER, the difference in PSNR could reach above 5 dB, but at low BER, the difference is at most 2 dB. The main reason for this is that the high BER, which requires the use of small packet sizes, produces a large number of COPs. Therefore, the code redundancies produced by the MultiCOP assignment differ much from those produced by the UniCOP assignment at high error rates. Accordingly, the MultiCOP assignment strategy is particularly very appealing for channels with high error rates, such as wireless channels.

Finally, the results show that for any given BER ϵ, the improvement in PSNR diminishes as the packet size decreases. This is due to the fact that fewer packets are lost since a decrease in packet size reduces packet loss probability. In turn, the likelihood of recovering source elements for both MultiCOP and UniCOP assignment strategies increases, resulting in similar overall expected utility. Of course, applications do not generally have the freedom of selecting packet sizes. The use of small packets increases the amount of the packet overhead, a fact which is not taken into account in the results presented here.

7. CONCLUSIONS

Although PET provides an excellent framework for optimal protection of scalable data sources against erasure, it suffers from a difficulty that all channel codes must span the entire collection of network packets. In many practical applications, the size of the data source is large and packet sizes must be relatively small, leading to the need for long and computationally demanding channel codes. Two solutions to this problem present themselves immediately. Small network packets can be concatenated forming larger packets, thereby reducing the codeword length of the channel codes. Unfortunately, an erasure channel model is required such that the larger packets must be considered lost if any of their constituent packets are lost. Clearly, this solution is unsuitable for channels with significant packet loss probability.

As an alternative, the code redundancy assignment optimized for the PET framework could be used with shorter channel codes representing smaller COPs. When data must be divided up into independently coded COPs with shorter channel codes, the MultiCOP assignment strategy proposed in this paper provides significant improvements in the expected utility (PSNR). Nevertheless, the need to use multiple COPs imposes a penalty of its own. One drawback of the multiple COP assignment strategy is the amount of computation required to determine optimal code redundancies at the transmitter. This is particularly significant when there are large numbers of source elements and/or the COP size is small. It is reasonable to expect exactly these conditions when transmitting a large compressed image over a wireless network. The development of fast algorithms for finding the MultiCOP assignment remains an active topic of investigation.

Including the codeword length as a parameter in the code redundancy assignment problem allows for flexibility in the choice of channel coding complexity. Since the channel decoder is generally more complex than the channel encoder, selecting short codewords will ease the computational burden at the receiver. This is particularly important for wireless mobile devices which have tight power and hence computation constraints.

REFERENCES

[1] J. M. Shapiro, "An embedded hierarchical image coder using zerotrees of wavelet coefficients," in *Proc. IEEE Data Compression Conference*, pp. 214–223, Snowbird, Utah, USA, March–April 1993.

[2] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.

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

[4] W. Li, "Overview of fine granular scalability in MPEG-4 video standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, 2001.

[5] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1737–1744, 1996.

[6] R. Puri and K. Ramchandran, "Multiple description source coding using forward error correction codes," in *Proc. 33rd Asilomar Conference on Signals, Systems, and Computers*, vol. 1, pp. 342–346, Pacific Grove, Calif, USA, October 1999.

[7] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degredation of image quality over packet erasure channels through forward error correction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 819–828, 2000.

[8] A. Mohr, R. Ladner, and E. Riskin, "Approximately optimal assignment for unequal loss protection," in *Proc. IEEE International Conference on Image Processing*, vol. 1, pp. 367–370, BC, Canada, September 2000.

[9] T. Stockhammer and C. Buchner, "Progressive texture video streaming for lossy packet networks," in *Proc. 11th Internatinal Packet Video Workshop*, Kyongju, Korea, May 2001.

[10] S. Dumitrescu, X. Wu, and Z. Wang, "Globally optimal uneven error-protected packetization of scalable code streams," in *Proc. IEEE Data Compression Conference*, pp. 73–82, Snowbird, Utah, USA, April 2002.

[11] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," in *Proc. IEEE International Conference on Image Processing*, vol. 2, pp. 165–168, Rochester, NY, USA, September 2002.

[12] J. Thie and D. Taubman, "Optimal protection assignment for scalable compressed images," in *Proc. IEEE International Conference on Image Processing*, vol. 3, pp. 713–716, Rochester, NY, USA, September 2002.

[13] D. Taubman and M. Marcellin, Eds., *JPEG2000: Image Compression Fundamentals, Standards and Practice*, vol. 642, Kluwer Academic Publishers, Boston, Mass, USA, 2001.

Johnson Thie received his B.E. degree in electrical engineering and his Master of Biomedical Engineering from the University of New South Wales, Australia, in 2000. He is currently a Ph.D. student in the School of Electrical Engineering & Telecommunications at the same university. Mr. Thie spent two summers (1996–1998) at the St George Hospital, Sydney, for industrial training. During the summer of 1998–1999, he worked at the Centre of Telecommunications and Industrial Physics, a division of Commonwealth Scientific & Industrial Research Organization (CSIRO), Australia. His research interests are distribution of compressed media over packet-based networks and general applications of signal/image processing in medicine.

David Taubman received his B.S. degree in computer science and mathematics and the B.E. degree in electrical engineering in 1986 and 1988 from the University of Sydney, and his M.S. and Ph.D. degrees in electrical engineering in 1992 and 1994 from the University of California, Berkeley. From 1994 to 1998, he worked at Hewlett-Packard's Research Laboratories in Palo Alto, California. He joined the University of New South Wales in 1998 as a Senior Lecturer in the School of Electrical Engineering and Telecommunications. Dr. Taubman was awarded the University Medal from the University of Sydney, as well as the Institution of Engineers, Australia Prize, and the Texas Instruments Prize for digital signal processing, all in 1998. He has received two Best Paper Awards from the IEEE Circuits and Systems Society in 1997 for the 1996 paper, "A common framework for rate and distortion based scaling of highly scalable compressed video," and from the IEEE Signal Processing Society in 2002 for the 2000 paper, "High performance scalable image compression with EBCOT." Dr. Taubman's research interests include highly scalable image and video compression, inverse problems in imaging, perceptual modeling, and multimedia distribution. He currently serves as an Associate Editor for the IEEE Transactions on Image Processing.

EURASIP Journal on Applied Signal Processing 2004:2, 220–235

Performance and Complexity Co-evaluation of the Advanced Video Coding Standard for Cost-Effective Multimedia Communications

Sergio Saponara
Interuniversity Microelectronics Center (IMEC), Kapeldreef 75, B-3001 Leuven, Belgium
Email: s.saponara@iet.unipi.it

Kristof Denolf
Interuniversity Microelectronics Center (IMEC), Kapeldreef 75, B-3001 Leuven, Belgium
Email: denolf@imec.be

Gauthier Lafruit
Interuniversity Microelectronics Center (IMEC), Kapeldreef 75, B-3001 Leuven, Belgium
Email: lafruit@imec.be

Carolina Blanch
Interuniversity Microelectronics Center (IMEC), Kapeldreef 75, B-3001 Leuven, Belgium
Email: blanch@imec.be

Jan Bormans
Interuniversity Microelectronics Center (IMEC), Kapeldreef 75, B-3001 Leuven, Belgium
Email: bormans@imec.be

Received 22 November 2002; Revised 26 September 2003

The advanced video codec (AVC) standard, recently defined by a joint video team (JVT) of ITU-T and ISO/IEC, is introduced in this paper together with its performance and complexity co-evaluation. While the basic framework is similar to the motion-compensated hybrid scheme of previous video coding standards, additional tools improve the compression efficiency at the expense of an increased implementation cost. As a first step to bridge the gap between the algorithmic design of a complex multimedia system and its cost-effective realization, a high-level co-evaluation approach is proposed and applied to a real-life AVC design. An exhaustive analysis of the codec compression efficiency versus complexity (memory and computational costs) project space is carried out at the early algorithmic design phase. If all new coding features are used, the improved AVC compression efficiency (up to 50% compared to current video coding technology) comes with a complexity increase of a factor 2 for the decoder and larger than one order of magnitude for the encoder. This represents a challenge for resource-constrained multimedia systems such as wireless devices or high-volume consumer electronics. The analysis also highlights important properties of the AVC framework allowing for complexity reduction at the high system level: when combining the new coding features, the implementation complexity accumulates, while the global compression efficiency becomes saturated. Thus, a proper use of the AVC tools maintains the same performance as the most complex configuration while considerably reducing complexity. The reported results provide inputs to assist the profile definition in the standard, highlight the AVC bottlenecks, and select optimal trade-offs between algorithmic performance and complexity.

Keywords and phrases: video compression standards (MPEG AVC, H.26L, H.264), power-optimized multimedia coding, cost-effective design of multimedia systems, complexity analysis.

1. INTRODUCTION

New applications and services in the communication and computing technology mainly focus on the processing and transmission of multimedia contents with portable and personal access to the information. While the enabling technologies for speech, data, text, and audio are available today (allowing the widespread diffusion of mobile phones, MP3 music players, global positioning systems to name but a few), the management of video information represents a remaining

design challenge for its inherent high data rates and storage burdens. To cope with this issue, the advanced video codec (AVC), recently defined in a standardization effort of the ITU-T and ISO/IEC joint video team (JVT) [1, 2, 3, 4], promises both enhanced compression efficiency over existing video coding standards (H.263 [5], MPEG-4 Part 2 [6, 7]) and network friendly video streaming. The codec aims at both conversational (bidirectional and real-time videotelephony, videoconferencing) and nonconversational (storage, broadcasting, streaming) applications for a wide range of bitrates over wireless and wired transmission networks.

Like previous video coding standards [5, 6, 7], AVC is based on a hybrid block-based motion compensation and transform-coding model. Additional features improve the compression efficiency and the error robustness at the expense of an increased implementation complexity. This directly affects the possibility for cost-effective development of AVC-based multimedia systems and hence the final success of the standard. The scope of this paper is the exploration of the compression efficiency versus implementation cost design space to provide early feedbacks on the AVC bottlenecks, select the optimal use of the coding features, and assist the definition of profiles in the standard. The complexity analysis focuses on the data transfer and storage, as these are the dominant cost factors in multimedia system design for both software- and hardware-based architectures [8, 9, 10, 11, 12, 13, 14, 15, 16]. Memory metrics are completed by computational burden measures. A comparison of the new codec with respect to current video coding technology, in terms of both compression efficiency and implementation cost, is also provided.

The paper is organized as follows. After a review of known profiling methodologies for multimedia system design, Section 2 defines and motivates the analysis approach adopted throughout the paper. A description of the upcoming standard including both encoder and decoder architectures is addressed in Section 3. Section 4 describes the testbench environment. Section 5 presents the global results obtained for the codec in terms of compression efficiency, memory cost, and computational burden. Section 6 exploits a multiobjective analysis to select the optimal trade-off between algorithmic performance and implementation cost. Section 7 deals with the definition of profiles in the standard. Conclusions are drawn in Section 8.

2. PERFORMANCE AND COMPLEXITY EVALUATION METHODOLOGY

As sketched in Figure 1, the design flow of complex multimedia systems such as video codecs typically features two main steps: an algorithmic development phase followed by a system implementation process. The first step focuses on algorithmic performance (peak signal-to-noise ratio (PSNR), visual appearance, and bit rate). The algorithmic specification is typically released as a paper description plus a software verification model (often in C). Usually, the software model is not optimized for a cost-effective realization since its scope is mainly a functional algorithmic verification and the target platform is unknown. Moreover, in the case of multimedia standards such as ITU-T and ISO/IEC video codecs, the verification software models (up to 100.000 C-code lines [10]) are written in different code styles since they are the results of the combined effort of multiple teams. The second step of the design flow deals with the actual system realization starting from the paper and software specification. Only at this late stage, the true implementation complexity of the algorithm is known, which will determine the cost of the user's terminal and hence its success and widespread diffusion or not. If the initial cost specifications are not reached, the gained complexity information is used to re-enter the design flow making new actions at algorithmic and then implementation levels. This time-consuming loop ends only when the complexity meets the user's requirements.

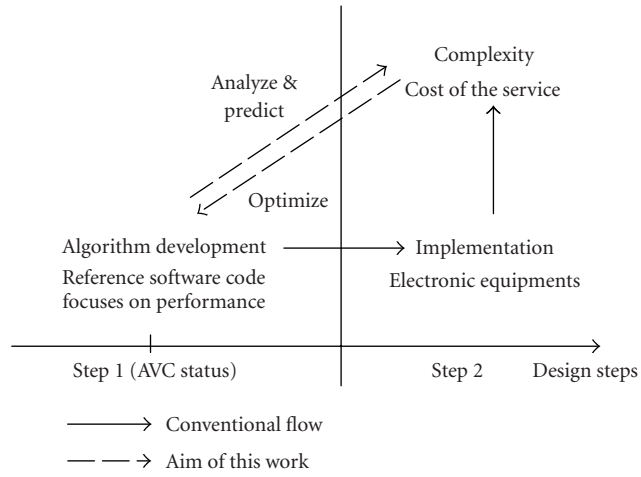


Figure 1: Algorithmic performance and complexity co-evaluation for cost-effective system design.

To bridge the gap between the algorithmic development of a new multimedia application and its cost-effective realization, we propose to explore the performance versus implementation cost design space at the early algorithmic design phase. The goal of this co-evaluation approach is twofold: (i) to assess the performance and implementation cost of a new multimedia system presenting also a comparison with current technology ("Analyze & Predict" arrow in Figure 1); (ii) to provide feedback on the realization bottlenecks and highlight the properties of the system allowing for complexity reduction at the early algorithmic design phase ("Optimize" arrow in Figure 1). This way, the time-consuming iterations of the conventional design flow can be avoided. Particularly, this paper focuses on the design of the AVC video coding standard for which a committee draft specification and a verification software C-model have been recently defined [1, 2]. The huge C-code complexity of multimedia systems makes an implementation cost analysis without additional help time consuming and error prone. Hence, a supporting framework for automated analysis of the executable software specification is essential

to apply the co-evaluation approach to a complex real-life design such as AVC. To this aim, the C-in-C-out ATOMIUM/Analysis environment [10, 15, 17] has been developed. It consists of a set of kernels providing functionality for data transfer and storage analysis and pruning. Using ATOMIUM involves three steps [10]: instrumenting the program, generating complexity data by executing the instrumented code with representative test stimuli, and postprocessing of this data.

High-level profiling analyses have been addressed in the past for previous ITU-T (H.263+ in [5]) and ISO/IEC (MPEG-4 Part 2 in [6, 18], MPEG-1/-2 decoder in [19]) video codecs. However, the above approaches focus mainly on computational complexity (processing time [5] or instruction-level [6, 19] profiling on a specific platform: typically general purpose CISC processors, e.g., Pentium in [5], or RISC processors, e.g., UltraSPARC in [6, 19]), while the actual implementation of H.263 and MPEG-4 codecs clearly demonstrates that multimedia applications are data dominated. As a consequence, data transfer and storage have a dominant impact on the cost-effective realization of multimedia systems for both hardware- and software-based platforms [8, 9, 10, 11, 12, 13, 14, 15, 16]. Application specific hardware implementations have the freedom to match the memory and communication architectures to the application. Thus, an efficient design flow exploits this to reduce area and power [8, 11, 12]. On the other hand, programmable processors rely on the memory hierarchy and on the communication bus architecture that come with them. Efficient use of these resources is crucial to obtain the required speeds as the performance gap between CPU and DRAM is growing every year [9, 13, 14, 15, 20]. This high-level analysis is also essential for an efficient hardware/software system partitioning. In [18], a complexity evaluation methodology based on the extraction of execution frequencies of core tasks is proposed. Combining this data with complexity figures for the core tasks on a specific platform, a performance estimate of the whole system on that platform is obtained. This approach relies on implementation cost measures already available for the single tasks (provided as benchmarks of a specific platform). Therefore, it is not suitable to analyze systems, such as AVC, featuring new algorithms for which complexity results are not available.

In this paper, the coding performance analysis is reported in terms of PSNR and bit rate, while the complexity metrics are the memory access frequency (total number of data transfers from/to memory per second) and the peak memory usage (maximum memory amount allocated by the source code) as counted within the ATOMIUM environment. These figures give a platform independent measure of the memory cost (storage and communication of data) and are completed with the processing time as a measure of the computational burden (processing time figures are measured on a Pentium IV at 1.7 GHz with Windows 2000). The software models used as input for this paper are the AVC JM2.1 [2] and the MPEG-4 Part 2 [7] (simple profile in [21]), both nonoptimized source codes.

3. ADVANCED VIDEO CODEC

3.1. Standard overview

An important concept of AVC is the separation of the system into two layers: a video coding layer (VCL), providing the high-compressed representation of data, and a network adaptation layer (NAL), packaging the coded data in an appropriate manner based on the characteristics of the transmission network. This study focuses on the VCL. For a description of NAL features, the reader is referred to [22, 23]. Figures 2 and 3 show the block diagram of the AVC decoder and encoder, respectively. In analogy with previous coding standards, the AVC final committee draft [1] does not explicitly define the architecture of the codec but rather it defines the syntax of an encoded video bitstream together with the decoding method. In practice, according to the structure of the AVC reference software [2], a compliant encoder and decoder are likely to include the functional tasks sketched in Figures 2 and 3. Nevertheless, particularly at the encoder side, there is space for variations in the sketched architecture to meet the requirements of the target application with the desired trade-off between algorithmic performance and cost. At the decoder side, the final architecture depends on the encoder profiles (i.e., combination of coding tools and syntax of the relevant bitstream) supported for decoding.

The framework defined in Figures 2 and 3 is similar to the one of previous standards: translational block-based motion estimation and compensation, residual coding in a transformed domain, and entropy coding of quantized transform coefficients. Basically, rectangular pictures can be coded in intra (I), inter (P), or bidirectional (B) modes. Both progressive and interlaced 4 : 2 : 0 YUV sequences are supported. Additional tools improve the compression efficiency, albeit at an increased implementation cost. The motion estimation and compensation schemes (ME and MC in Figures 2 and 3) support multiple previous reference pictures (up to 5) and a large number of different block sizes (from 16×16 up to 7 modes including 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 pixel blocks). The motion vector field can be specified with a higher spatial accuracy, quarter- or eighth-pixel[1] resolution instead of half pixel. Pixel interpolation is based on a finite impulse response (FIR) filtering operation: 6 taps for the quarter resolution and 8 taps for the eighth one. A rate-distortion (RD) Lagrangian technique [24] optimizes both motion estimation and coding mode decisions. Since the residual coding is in a transformed domain, a Hadamard transform can be used to improve the performances of conventional error cost functions such as the sum of absolute differences (SAD). Moreover, a deblocking filter within the motion compensation loop aims at improving prediction and reducing visual artifacts. AVC adopts spatial prediction for intracoding, being the pixels

[1] The eighth-pixel resolution, available in AVC JM 2.1 [1, 2], is no longer supported in the last release [3]. However, this tool is kept during the proposed analysis since it is useful for high-rate video applications (Section 7).

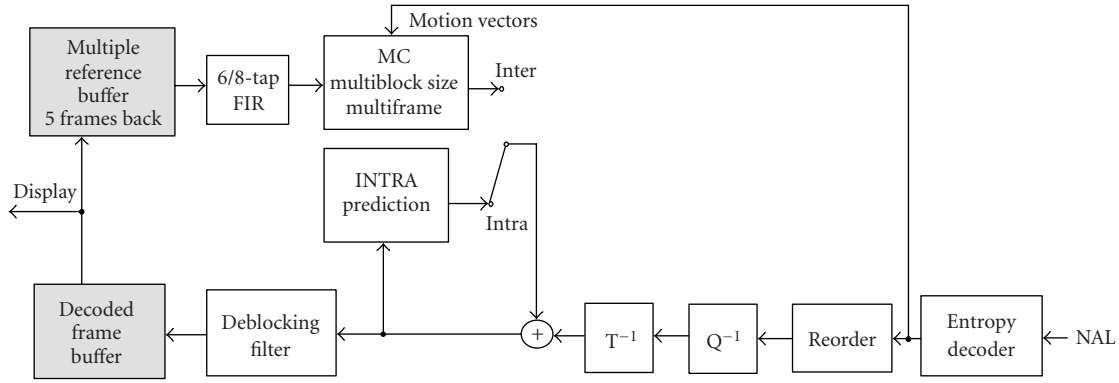


Figure 2: AVC decoder block diagram.

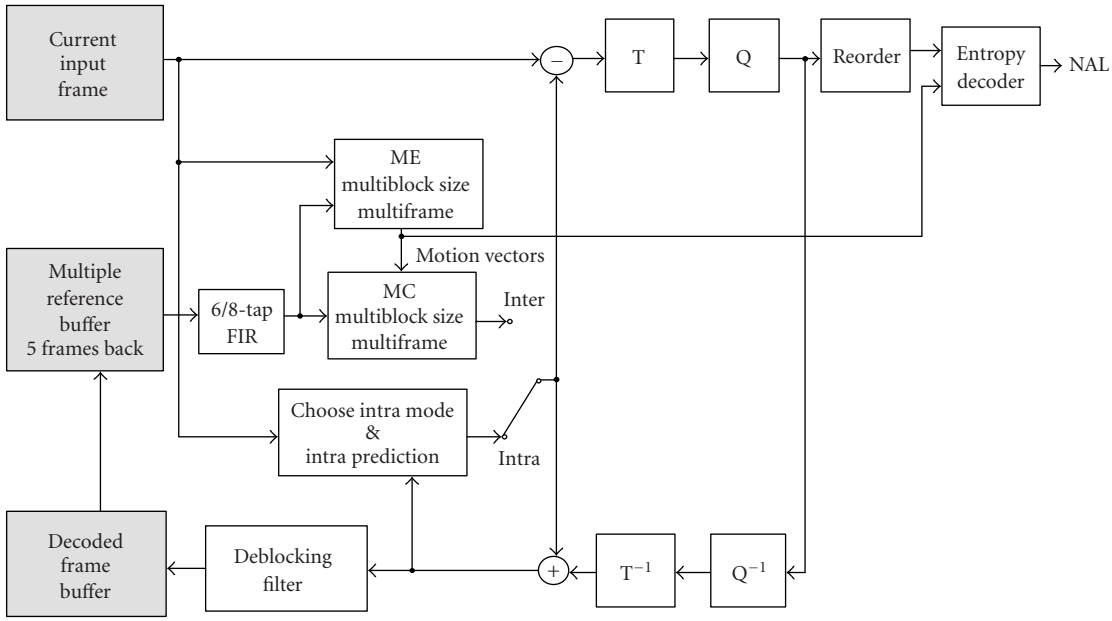


Figure 3: AVC encoder block diagram.

predicted from the neighbouring samples of already coded blocks. To this aim, the standard provides a DC plus 8 directional modes involving linear combinations of the samples. The conventional 8×8 floating-point discrete cosine transform, specified with rounding error margins, is replaced by a purely integer spatial transform (T and T^{-1} in Figures 2 and 3), basically working on 4×4 shapes. The small sizes help to reduce blocking artifacts while the integer specification prevents any mismatch between the encoder and the decoder. Finally, two methods are specified for entropy coding: a universal variable-length coder (UVLC) that uses a single reversible VLC table for all syntax elements and a more sophisticated context adaptive binary arithmetic coder (CABAC) [25].

3.2. Related work

Several contributions have recently been proposed to assess the coding efficiency of the AVC/H.26L scheme [3, 22, 25, 26, 27, 28] (H.26L is the original ITU-T project used as a starting point for the AVC standard, released as ITU-T H.264 and ISO/IEC MPEG-4 Part 10). Although this analysis covers all tools, the new features are typically tested independently comparing the performance of a basic configuration to the same configuration plus the tool under evaluation. In this way, the intertool dependencies and their impact on the trade-off between coding gain and complexity are not fully explored yet. Indeed, the achievable coding gain is greater for basic configurations, where the other tools are off and video data still feature a high correlation. For codec configurations

Table 1: Characteristics of the test sequences with their targeted bit rate.

Test case	Video sequence	Format (pixels)	Frame rate (Hz)	QP	Bit rate (Kbps)
MD	Mother & Daughter	QCIF (176 × 144)	30	18	40
FOR1	Foreman	QCIF (176 × 144)	25	17	150
FOR2	Foreman	CIF (352 × 288)	25	17	450
CM	Calendar & Mobile	CIF (352 × 288)	15	12	2000

in which a certain number of tools are already on, the residual correlation is lower and the further achievable gain is less noticeable.

Complexity assessment contributions have been proposed in [26, 27, 29, 30]. However, these works do not exhaustively address the problem since just one side of the codec is considered (the encoder in [26, 27] and the decoder in [29, 30]) and/or the analysis of the complete tool-set provided by the upcoming standard is not presented. Typically, the use of B-frames, CABAC, multireference frames, and eighth resolution is not considered. Consequently, the focus is mostly on a baseline implementation suitable for low-complexity and low-bit-rate applications (e.g., video conversation), while AVC aims at both conversational and nonconversational applications in which these discarded tools play an important role [3, 25, 28]. Furthermore, the complexity evaluation is mainly based on computational cost figures, while data transfer and storage exploration proved to be mandatory for efficient implementation of video systems [8, 9, 10, 11, 12, 13, 14, 15, 16] (see Section 2). Access frequency figures are reported in [29] for a H.26L decoder, but the analysis focuses on the communication between an ARM 9 CPU and the RAM, being a platform-dependent measure of the bus bandwidth rather than a platform-independent exploration of the system.

4. TEST ENVIRONMENT

4.1. Test sequences

The proposed testbench consists of 4 sequences with different grades of dynamism, formats, and target bit rates. Their characteristics are sketched in Table 1. Mother & Daughter 30 Hz QCIF (MD) is a typical head and shoulder sequence occurring in very low-bit-rate applications (tens of Kbps). Foreman 25 Hz QCIF (FOR1) has a medium complexity, being a good test for low bit rate applications ranging from tens to few hundreds of Kbps. The CIF version of Foreman (FOR2) is a useful test case for middle-rate applications. Finally, Calendar & Mobile 15 Hz CIF (CM) is a high-complexity sequence with lot of movements including rotation and is a good test for high-rate applications (thousands of Kbps). Since the current standard description does not provide online rate controls, the test sequences in Section 5.1 are coded with a fixed quantization parameter (QP in Table 2) to achieve the target bit rate. The dependency of the proposed analysis on the QP value is addressed in Section 5.2.

4.2. Test cases

The paper reports for each test video 18 different AVC configurations whose descriptions are shown in Table 2.

For each test case (identified by a number from 0 to 17), Table 2 details the activation status of the optional video tools with respect to a basic AVC configuration (case 0) characterized by a search range of 8, 1 reference frame, quarter-pixel resolution, intracoding by 9 prediction modes, in-loop deblocking, UVLC entropy coder, and a first I picture followed by all P pictures. The tools which are changing between two successive test cases are highlighted in bold style in Tables 2 and 3. Comparisons with MPEG-4 Part 2 [7], simple profile in [21] with a 16 search size, half-pixel resolution, and I and P pictures (referred to as test case M4 in the next sections) are provided in Sections 5 and 6.

The 18 reported AVC configurations are selected, for sake of space, as representatives of more than 50 considered test cases. The first two cases represent a "simple" AVC implementation with all new video tools off (with search displacements of 8 for case 0 and of 16 for case 1). Then, in cases 2 to 9 ("accumulative video tool enabling" in Table 2), the new AVC features are added one by one up to "complex" configurations, with all tools on (including B pictures with search displacements of 16 for cases 10 and 12, and of 32 for case 11), reaching the best coding performance although at maximum complexity. Comparing the test cases from 3 to 12 with the basic configurations 0 and 1 gives feedbacks about the coding efficiency versus complexity trade-off of the new AVC video tools. As it will be explained further, cases 13 to 17 in Table 2 have been properly selected to achieve roughly the same coding efficiency as the complex cases, while considerably reducing the complexity overhead by discarding some tools and reducing the number of reference frames and the search area. The overall set of AVC configurations (roughly 50) is the same for all the considered test sequences. As it will be detailed in Sections 5 and 6, the performance and usefulness of the different video tools depend on the considered bit rate and hence on the considered sequence (MD for tens of Kbps, FOR1 and FOR2 from tens to hundreds of Kbps, and CM for thousands of Kbps). During the selection, among the set of 50 configurations, of the 18 more representative test cases to be reported in this paper, the configurations from 0 to 12 ("simple," "accumulative video tool enabling," and "complex") have been chosen identical for all the video sequences, while the configurations from 13 to 17 (cost-efficient) feature some differences. Table 2 refers to FOR1 and FOR2, while Table 3 reports the cost-efficient configurations for MD and CM.

Table 2: AVC configurations cases—FOR1 and FOR2 test sequences (Y = tool on, N = tool off).

Case number	Simple		Accumulative video tool enabling								Complex			Cost-efficient				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Search range	8	**16**	16	16	16	16	16	16	16	16	16	**32**	**16**	16	**8**	8	8	8
Block sizes[a]	1	1	**4**	**7**	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Ref. frames	1	1	1	1	**3**	**5**	5	5	5	5	5	5	5	5	5	**3**	**2**	**1**
RD-Lagrange	*N*	*N*	*N*	*N*	*N*	*N*	**Y**	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*
Hadamard	*N*	*N*	*N*	*N*	*N*	*N*	*N*	**Y**	*Y*	*Y*	*Y*	*Y*	*Y*	**N**	*N*	*N*	*N*	*N*
Eighth resolution	*N*	*N*	*N*	*N*	*N*	*N*	*N*	*N*	**Y**	*Y*	*Y*	*Y*	*Y*	**N**	*N*	*N*	*N*	*N*
CABAC	*N*	*N*	*N*	*N*	*N*	*N*	*N*	*N*	*N*	**Y**	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*
B-frames	*N*	*N*	*N*	*N*	*N*	*N*	*N*	*N*	*N*	*N*	**1**	1	**2**	**1**	1	1	1	1

Table 3: AVC "cost-efficient" configurations cases—MD and CM test sequences (Y = tool on, N = tool off).

Case number	MD					CM				
	13	14	15	16	17	13	14	15	16	17
Search range	16	16	16	16	**8**	8	8	8	8	8
Block sizes[a]	7	7	7	7	7	7	7	7	7	7
Ref. frames	2	**1**	1	1	1	5	**3**	**2**	**3**	**2**
RD-Lagrange	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	**N**	*N*
Hadamard	*Y*	*Y*	*Y*	**N**	*N*	*Y*	**N**	*N*	*N*	*N*
Eighth resolution	*Y*	*Y*	**N**	*N*	*N*	*Y*	*Y*	*Y*	*Y*	*Y*
CABAC	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*
B-frames	*N*	*N*	*N*	*N*	*N*	1	1	1	1	1

[a]If block sizes = 1, only the 16 × 16 mode is on; if block sizes = 4, then 16 × 16, 16 × 8, 8 × 16, and 8 × 8 modes are on; otherwise, all modes are on.

5. AVC PERFORMANCE AND COMPLEXITY ASSESSMENT

5.1. Codec analysis

An overview of the encoder and decoder results (PSNR-Y, bit rate, peak memory usage, memory access frequency, processing time) for the 18 AVC test cases and the M4 one is summarized in Figures 4, 5, 6, 7, 8, 9, 10, and 11 and Tables 4, 5, and 6.

Coding performance results

Figures 4, 5, 6, and 7 list the rate-distortion results for all the video inputs using the fixed QP values reported in Table 1. For the sake of clarity, a rhombus represents the simple AVC configurations (cases 0 and 1), a cross identifies test cases from 2 to 9, a square represents complex AVC configurations (cases 10 to 12), a triangle indicates the cost-efficient configurations (cases 13 to 17 in Tables 2 and 3), and a circle refers to M4 results.

Clearly, AVC is a new codec generation featuring an outstanding coding efficiency: if all the novel video tools are used, AVC leads to an average 40% bit saving plus a 1–2 dB PSNR gain compared to previous M4 video coding standard (see results for test cases 10, 11, and M4 in Figures 4, 5, 6, and 7).

Complexity results

Figures 8, 9, 10, and 11 deal with processing time and memory access frequency costs for both AVC encoder (Figures 8 and 9) and decoder (Figures 10 and 11). In these figures, for all video inputs, the reported values are normalized with respect to the ones of the relevant test cases 0. A close similarity between the processing time and the memory access frequency curves emerges from the comparison of Figures 8 and 9 at the encoder and Figures 10 and 11 at the decoder. Moreover, the analysis of the performance and complexity metrics shows that the new coding scheme acts likewise for all input sequences, particularly at middle and low bit rates (see the behaviors of MD, FOR1, and FOR2 in Figures 4, 5, 6, 8, 9, 10, and 11). Small differences arise for high rate video applications (CM) as emerges from Figures 10 and 11.

Absolute complexity values are reported in Table 4 listing the range achieved by the different AVC configurations (rows Min and Max) and the complexity results of M4 as a reference. The processing time values in Table 4 are expressed in a relative way: they refer to the time needed to encode/decode on a Pentium IV at 1.7 GHz, 1 second of the original test sequence, that is to say, (see Table 1) 25 frames of FOR1 and FOR2, 30 frames of MD, and 15 frames of CM. As a consequence, meting real-time constraints entails a relative processing time smaller than 1.

The encoder peak memory usage depends on the video format and linearly on the number of reference frames and the search size. The influence of the other coding tools and the input video characteristics is negligible. At the decoder side, the peak memory usage depends only on the video format and on the maximum number of reference frames to

Table 4: Codec complexity for MPEG-4 Part 2 (M4) and AVC codec (ranges Min, Max).

Test cases		Encoder			Decoder		
		Relative time	Accesses (10^9/s)	Peak memory (MB)	Relative time	Accesses (10^6/s)	Peak memory (MB)
MD	AVC Min	6.48	1.40	2.19	0.30	61.10	1.06
	AVC Max	409.12	79.92	15.60	0.69	104.41	1.38
	M4	4.15	1.32	2.97	0.49	102.79	1.30
FOR1	AVC Min	5.40	1.18	2.19	0.58	90.16	1.06
	AVC Max	330.87	65.78	15.60	1.30	153.88	1.38
	M4	3.55	1.15	2.97	0.61	107.51	1.30
FOR2	AVC Min	21.70	4.60	7.31	3.04	385.21	2.91
	AVC Max	1117.48	258.01	26.87	5.51	636.90	4.15
	M4	14.64	4.96	9.88	2.25	411.4	3.13
CM	AVC Min	12.98	2.75	7.31	2.33	287.10	2.91
	AVC Max	567.37	134.26	26.87	4.26	492.54	4.15
	M4	11.32	3.19	9.88	1.76	278.30	3.13

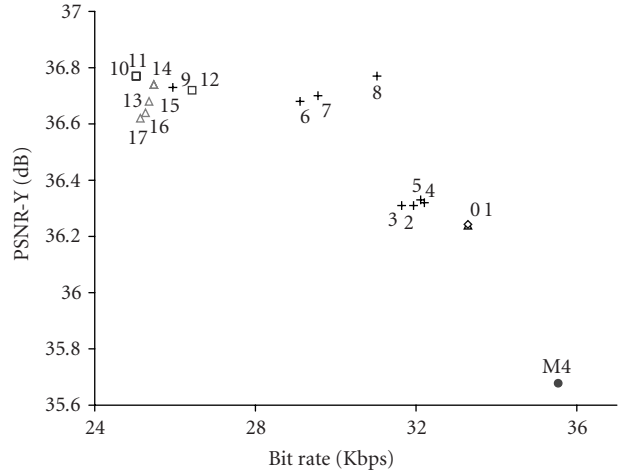


Figure 4: Rate-distortion results for the MD test cases.

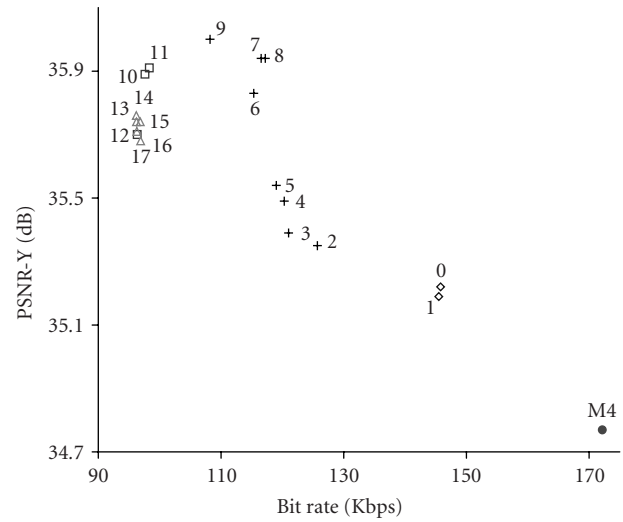


Figure 5: Rate-distortion results for the FOR1 test cases.

Table 5: Decoder memory usage (MB) versus video format and maximum number of decodable reference frames.

	1 frame	2 frames	3 frames	4 frames	5 frames
QCIF	1.06	1.14	1.22	1.30	1.38
CIF	2.91	3.22	3.53	3.84	4.15

decode. Peak memory usage dependencies for the decoder and the encoder are detailed in Tables 5 and 6.

To better highlight the intertools dependencies, the complexity results of Figures 8, 9, 10, and 11 and Tables 4, 5, and 6 refer to the whole AVC coder and decoder. A functional access and time distribution over the different components (e.g., motion estimator, intra predictor, etc.) have already been addressed by the authors in [31] for simple and complex configurations. At the encoder side, up to 90% of the complexity is due to motion estimation. The decoder's main bottlenecks are the motion compensation (up to 30% and 60% for simple and complex configurations, respectively) and the intrareconstruction (nearly 20% and 15% for simple and complex configurations, respectively). With respect to previous ITU-T and ISO/IEC standards, another important component of the AVC decoder is the in-loop deblocking filter (see further details in Section 5.3) whose implementation entails an overhead up to 6% for the access frequency and 10% for the processing time.

Analysis of coding performance and complexity results

AVC is a new codec generation featuring an outstanding coding efficiency, but its cost-effective realization is a big challenge. If all the novel coding features are used, AVC leads to an average 40% bit saving plus a 1–2 dB PSNR gain compared to previous video coding standards (see results for test cases 10, 11, and M4 in Figures 4, 5, 6, and 7). In this way, it represents the enabling technology for the widespread diffusion of

Table 6: Encoder memory usage (MB) versus reference frame number, search size, and video format.

Search size	QCIF			CIF		
	1 frame	3 frames	5 frames	1 frame	3 frames	5 frames
32	5.76	10.69	15.60	10.87	18.87	26.87
16	2.92	5.09	7.14	8.03	13.06	18.41
8	2.19	3.64	4.98	7.31	11.94	16.25

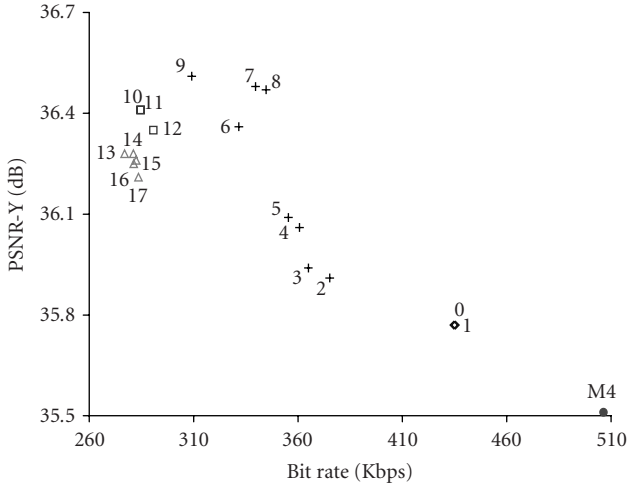


Figure 6: Rate-distortion results for the FOR2 test cases.

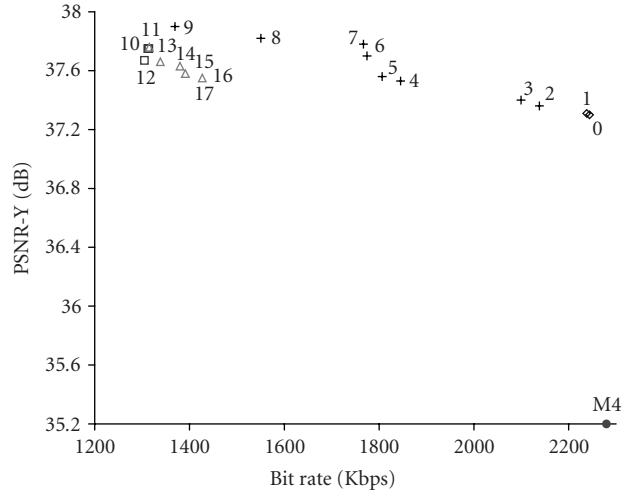


Figure 7: Rate-distortion results for the CM test cases.

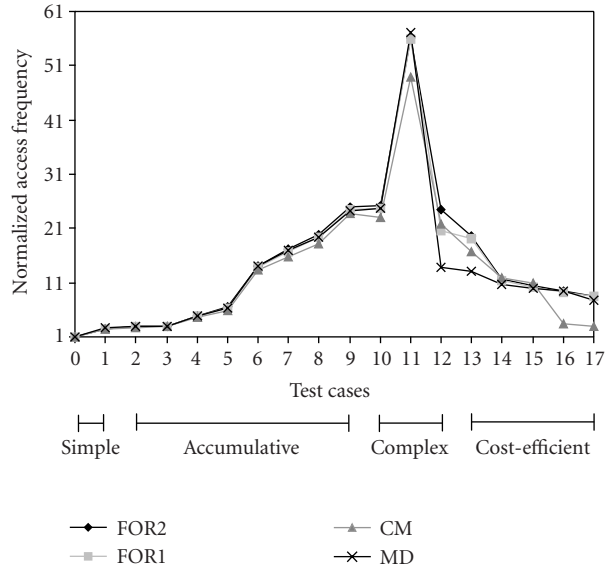


Figure 8: Normalized access frequency in case of AVC encoder.

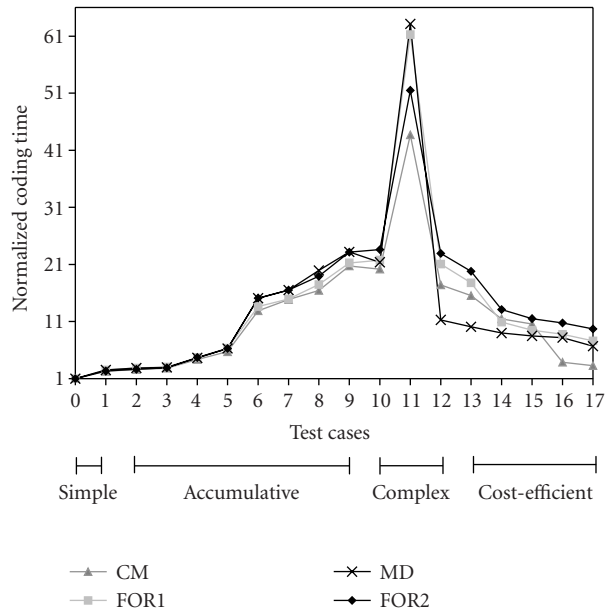


Figure 9: Normalized AVC coding time.

multimedia communication over wired and wireless transmission networks such as xDSL, 3G mobile phones, and WLAN. However, these figures come with a memory and computational complexity increase of more than one order of magnitude at the encoder. The decoder's complexity increase amounts to a factor 1.5–2 (see results for test cases 0, 10, and 11 in Figures 8, 9, 10, and 11 and those for the AVC Max and M4 rows in Table 4). These increase factors are higher for the lower bit rate video as it emerges from the comparison of cases 0 and 11 in Figures 8, 9, 10, and 11 for the different video inputs. Case 11, for which the maximum

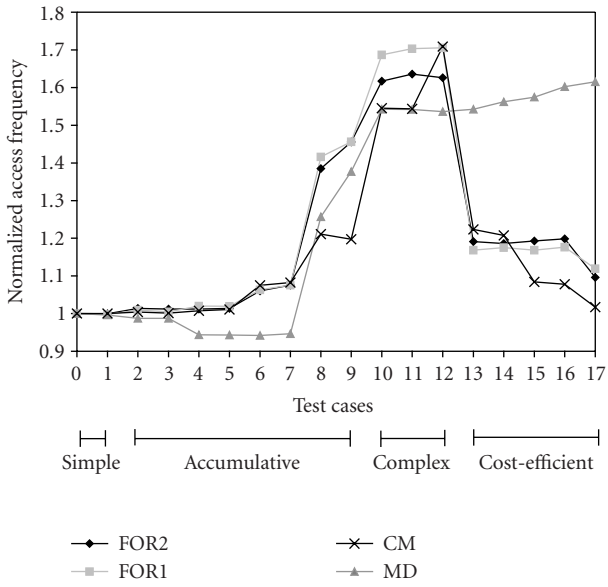


Figure 10: Normalized access frequency in case of AVC decoder.

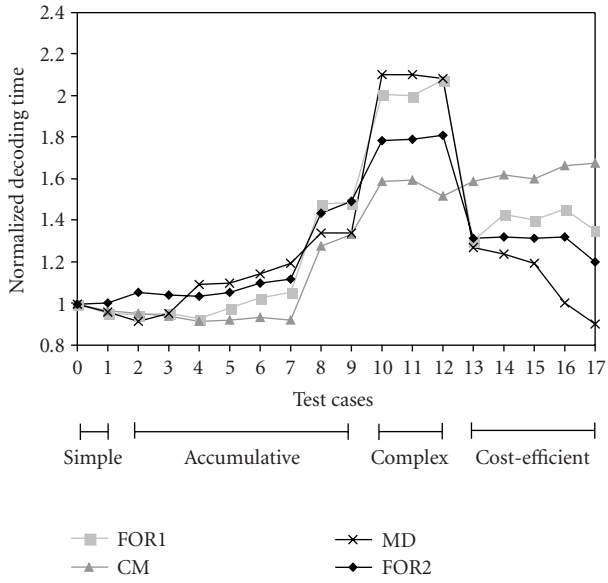


Figure 11: Normalized AVC decoding time.

complexity is measured, is the configuration used in [3] to show the AVC compression efficiency with respect to previous video coding standards. Finally, the complexity ratio between the encoder and the decoder further highlights the AVC bottleneck, particularly for conversational applications (e.g., videotelephony), where both the encoder and the decoder capabilities must be integrated in the user's terminal. For a simple profile, Min rows in Table 4, the encoder requires an access frequency and coding time at least 10 times that of the decoder and uses 2 times more memory space. For complex profiles, Max rows in Table 4, the encoder access frequency is two orders of magnitude larger than the decoder one, while the peak memory usage is one order of magnitude higher.

The above measurements refer to nonoptimized source code and hence the future application of algorithmic and architectural design optimizations will lead to a decrease of the absolute complexity values, as it is the case in implementations of previous ITU-T and ISO/IEC standards [5, 11, 15, 16]. For instance, [32] recently proposed a fast motion estimation technique exploiting the new features of AVC, such as multireference frames and variable block sizes. The authors report a complexity reduction of a factor 5–6 with respect to a nonoptimized encoder realization based on the full search. However, the large complexity ratio between the reference codes of AVC and M4 (one order of magnitude at the encoder and a factor 2 at the decoder) presents a serious challenge requiring an exhaustive system exploration starting from the early standard design phase. Indeed, the performance growth rate predicted by Moore's law for the CPU amounts roughly to a factor 2 every 18 months. If we assume the same optimization factor as previously achieved for M4 to the current AVC code, but without any further system-level investigation, a cost-effective implementation could still not be scheduled before 2007 (i.e., the algorithmic complexity increase at the encoder would be covered in about four years and a half by the silicon technology improvements). Taking into account the lower performance growth rate of memories compared to CPU [20], the above time figure would even be worse.

The results in Figures 4, 5, 6, 7, 8, 9, 10, and 11 also provide useful hints for the selection of the optimal trade-off between coding efficiency and implementation complexity in order to maximize the utility for the final user. Indeed, the analysis of the above data clearly demonstrates a property of the AVC scheme: when combining the new coding features, the relevant implementation complexity accumulates (see the waveforms in Figures 8, 9, 10, and 11 for the test cases 0 to 11), while the global compression efficiency saturates (see the clusters in Figures 4, 5, 6, and 7 for the test cases 9 to 17). As a matter of fact, the achievable coding gain when enabling one of the new AVC features is greater for basic codec configurations, where the other tools are off and video data still feature a high correlation. For codec configurations in which a certain number of tools are already on, the residual data correlation is lower and hence, the further achievable gain is less noticeable, that is, the global compression efficiency saturates.

As a consequence, a "smart" selection of the new coding features can allow for roughly the same performances as a complex one (all tools on) but with a considerable complexity reduction. The coding efficiency (Figures 4, 5, 6, and 7) of test cases 13 to 17 is similar to that of cases 10 and 11, but their implementation cost (Figures 8, 9, 10, and 11) is closer to the basic cases 0 and 1. The achievable saving factor is at least 6.5 for the encoder. At the decoder side, the range of variation among simple and complex configurations is smaller, therefore, the saving is less noticeable than for the encoder. No complexity reduction is achieved for high rate video (CM), while saving factors of roughly 1.5 for both time

and memory metrics can be achieved for low bit rate videos. A single AVC configuration able to maximize coding efficiency, while minimizing memory and computational costs, does not exist. However, different configurations leading to several performance/cost trade-offs exist. To find these configurations, and hence to highlight the bottlenecks of AVC, a multiobjective optimization problem (solved, as it will be explained further, through a Pareto curve analysis) is addressed in Section 6 to explore the five-dimensional design space of PSNR, bit rate, computational burden, memory access frequency, and storage.

5.2. Performance and complexity analysis versus QP

Typically, video codecs incorporate a rate control scheme to target a given bit rate by adapting the quantization level. Since the standard description used in this paper does not yet provide such regulator, this section details the impact of different QP values on the analysis addressed in Section 5.1. All measurements described above are repeated on the 4 test sequences using several QP values (12, 16, 20, 24, 28) next to the fixed ones set in Table 1. To be noted that this analysis refers to the QP range defined in the JM2.1 implementation of the standard. Figure 12 sketches rate-distortion results: the points with higher PSNR and bit rate values are obtained with lower QP values. Figures 13, and 14 present the encoder and decoder complexity metrics expressed in terms of memory access frequency and processing time (expressed as relative time like in Section 5.1). In Figures 13 and 14, an arrow indicates the direction of growing QP values and hence decreasing bit rates. All these figures refer to the FOR2 video, covering a range from 100 to 1100 Kbps. Four representative AVC configurations are considered: cases 0, 10, 11, and 17.

The rate-distortion results (Figure 12) show a typical logarithmic behavior. For all bit rates, the complex configurations (cases 10 and 11) achieve at least a 2 dB PSNR increment versus the simple one (case 0). As expected from Section 5.1, the coding performances with a search size of 16 and 32 are practically the same. With respect to the M4 standard, the same PSNR results are achieved with a 50% reduced bit rate enabling full-rate video communication over today wireless and wired networks. For instance, according to the results of Figure 12, a complex CIF video like Foreman can be transmitted at 25 Hz and 36 dB with less than 300 Kbps being compatible with 3G wireless network capabilities. The analysis of Figure 12 for the whole bandwidth range further highlights the importance of AVC: even in case of broadband networks (e.g., xDSL and WLAN), nowadays multimedia communication terminals, based on MPEG-4 Part 2 and H.263 technologies, lead to a video coding and transmission 3 dB poorer on the same bit rate or they double the bandwidth (thus increasing the cost of the service) required to reach a certain PSNR level. Moreover, the high coding efficiency of AVC allows the insertion of some redundancy in the source coder to improve the transmission robustness in error-prone channels [22, 23, 33].

As already shown in Section 5.1, a proper use of the AVC tools allows for nearly the same efficiency as the complex cases with a considerable complexity reduction. Indeed, while in complexity figures (Figures 13 and 14) the implementation cost of case 17 is close to the simple one, in Figure 12 the relevant coding efficiency results are close to the complex ones (the difference between case 17 and complex curves is below 0.4 dB and for QP $\geq$ 16 the same results are achieved). The encoder data transfer and processing time practically do not depend on QP: indeed for each test case in Figure 13, the points with different QP values show nearly the same coding time and access frequency. At the decoder side (Figure 14), this dependency is more noticeable: as expected from literature [30], the higher the QP value (and hence the lower the bit rate), the lower the complexity. Finally, the storage requirements at both the encoder and the decoder (Figure 15) sides are not affected by the selected QP.

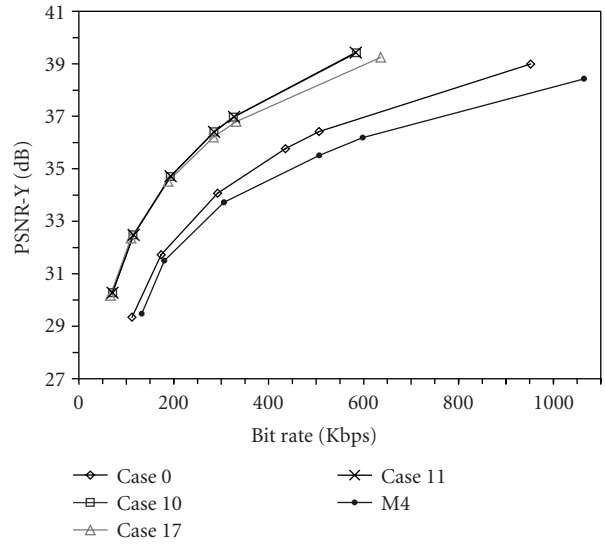


Figure 12: PSNR-Y (dB) versus bit rate in case of FOR2.

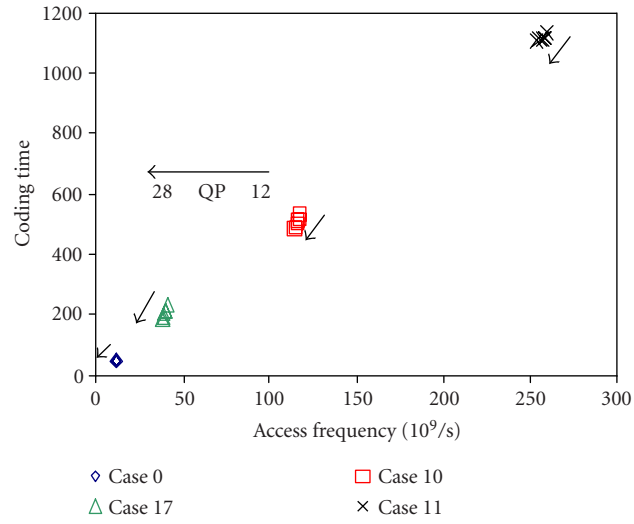


Figure 13: Encoder complexity for various QP values.

The dependency of AVC performance and complexity on the QP value has also been analyzed for the other test videos at lower (MD, FOR1) and higher (CM) rates. The achieved results are similar to those presented for FOR2.

5.3. Low-complexity AVC configuration

To test some low-complexity configurations not included in the basic scheme (half-pixel accuracy instead of quarter one and without in-loop deblocking), the JM2.1 code was suitably modified. Results for the same test cases of previous sections prove that restricting to half-pixel resolution decreases the compression efficiency (up to 30%, particularly for complex video inputs). Reducing the pixel accuracy can be useful only for very low rate video (MD) coded with a complex AVC profile. In this case, the lower pixel accuracy does not affect the coding efficiency and allows for a complexity reduction (both access frequency and processing time) of 10 and 15% for the encoder and decoder, respectively. As concerns deblocking, its use leads to PSNR (up to 0.7 dB) and bit rate (up to 6% saving) improvements. The complexity overhead is negligible at the encoder side and is up to 6% (access frequency) and 10% (processing time) at the decoder side. As proved in literature [13], the PSNR analysis is not enough for a fair assessment of the deblocking tool since a subjective analysis is also required. The latter, in addiction to the above rate-distortion gain, confirms the effectiveness of the insertion of deblocking within the basic standard profile [1].

6. PERFORMANCE VERSUS COST TRADE-OFF USING PARETO ANALYSIS

As shown in Section 5.1, achieving a good balance between algorithmic performance (coding efficiency) and cost (memory and computational complexity) is the first step to address the challenge of a cost-effective AVC realization. A Pareto curve [9, 34, 35] is a powerful instrument to select the right trade-off between these conflicting issues at system level. In a search space with multiple axes, it only represents the potentially interesting points and excludes all the others having an equally good or worse solution for all the axes.

The multi objective design space exploration is reported in this section for the FOR2 (see Figures 16, 17, 18, and 19) and CM (see Figures 20, 21, 22, and 23) video inputs. The algorithmic performance is measured as the required bit rate achieving a fixed PSNR (36 dB for the target FOR2 video, covering a range from 250 to 500 Kbps, and 37.6 dB for the CM video, covering a range from 1000 to 3000 Kbps). The Pareto analysis has also been applied to the other video inputs at very low bit-rates (MD, covering a 20–50 Kbps range) and low bit rates (FOR1, covering a 80–200 Kbps range) achieving similar results to those obtained for the FOR2 video.

Figures 16 and 17 sketch the FOR2 sequence Pareto curves for the encoder using as cost metrics the memory access frequency and the peak memory usage. Figures 18 and 19 show the same analysis for the decoder. A rhombus represents the simple AVC configurations (cases 0 and 1), a cross refers to test cases from 2 to 9, a square represents complex AVC configurations (cases 10 to 12), a triangle indicates the cost-efficient configurations (cases 13 to 17), and finally a circle identifies the M4 results. A Pareto analysis for the processing time is not presented since it is linear with the access frequency (see Figures 8, 9, 10, 11, 13, and 14), leading to the same conclusions.

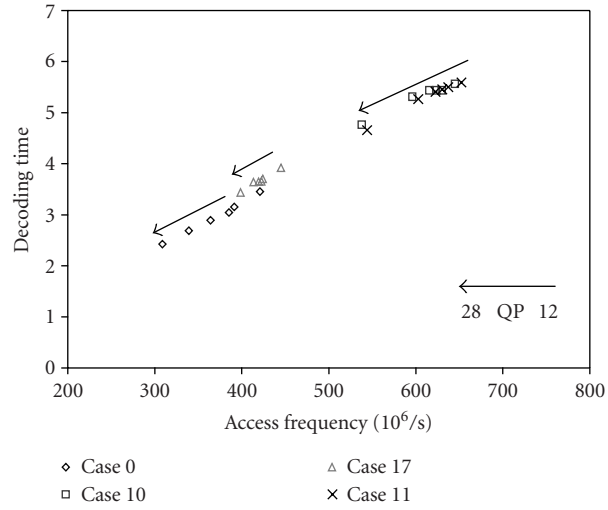


Figure 14: Decoder complexity for various QP values.

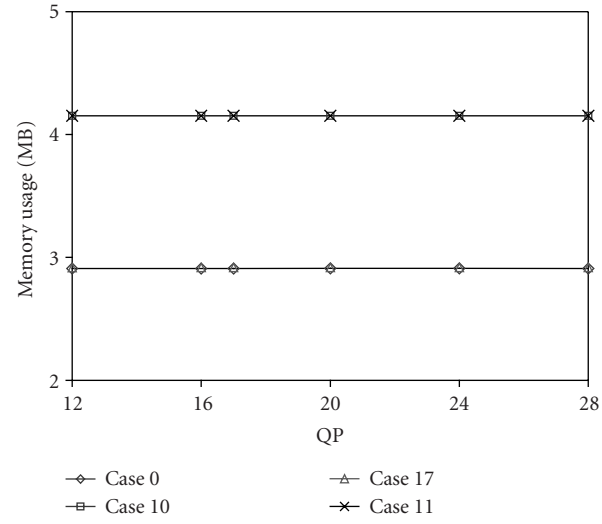


Figure 15: Decoder memory usage versus QP.

A simple AVC configuration (case 0) outperforms M4 since a lower bit rate (greater performance) is achieved for the same costs. Among the 18 tests, cases 4, 5, 6, 7, 8, 9, 10, 11, 12 are not interesting (namely, the above Pareto curves) since they offer a certain coding performance at a higher cost with respect to points 0, 1, 2, 3, 13, 14, 15, 16, 17 near the Pareto curves. The latter points offer different optimal trade-offs. Case 0 is the less complex and 13 is the most performing in coding efficiency. The results at low (FOR1 test video) and very low (MD test video) bit rates lead to similar observations as the middle-rate ones achieved in the FOR2 analysis.

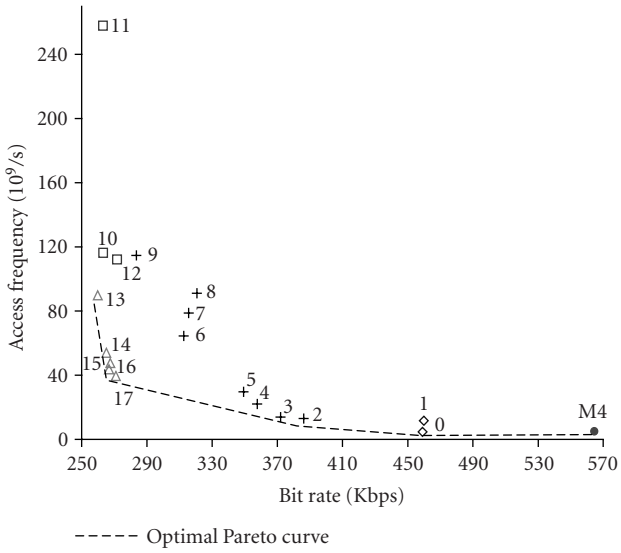


Figure 16: Encoder analysis, data transfer as cost in case of FOR2.

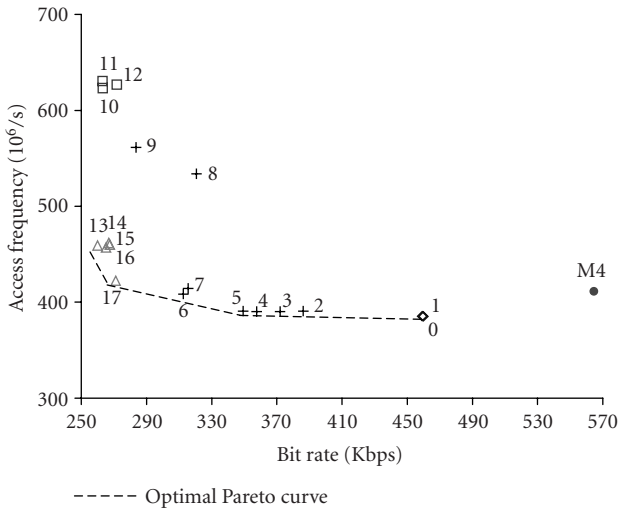


Figure 18: Decoder analysis, data transfer as cost in case of FOR2.

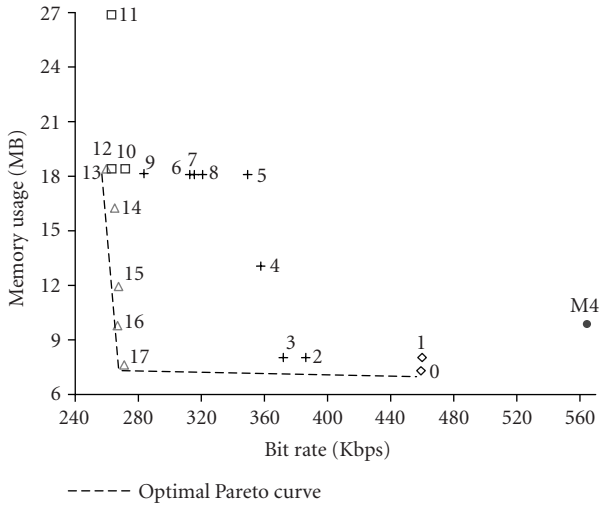


Figure 17: Encoder analysis, data storage as cost in case of FOR2.

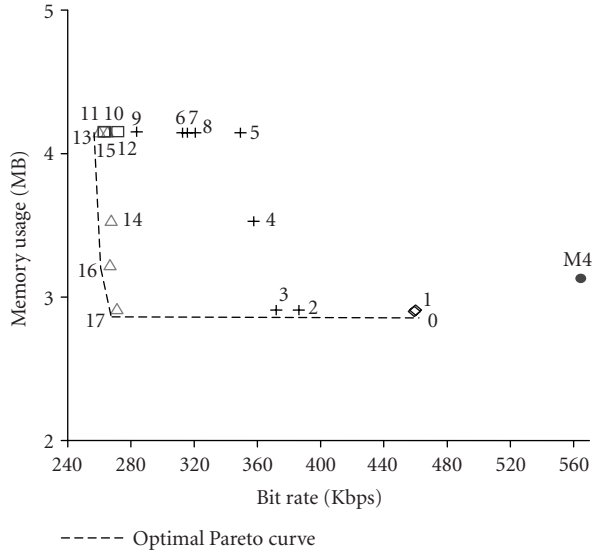


Figure 19: Decoder analysis, data storage as cost in case of FOR2.

The above analysis presents some differences when applied to high-rate video applications. With reference to the CM test, Figures 20 and 21 sketch the Pareto curves for the encoder using as cost metrics the memory access frequency and the peak memory usage. Figures 22 and 23 show the same analysis for the decoder. Differently from the results of Figures 16, 17, 18, and 19, in Figures 20, 21, 22, and 23, complex configurations such as cases 9, 10, and 12 are near the Pareto optimal curves.

From the combined analysis of the Pareto plots (Figures 16, 17, 18, 19, 20, 21, 22, and 23) and their description in Tables 2 and 3 and Section 5.1, the following considerations can be derived, valid for all kind of sequences (both low and high bit rates).

(i) The main AVC bottleneck is the combination of multiple reference frames and large search sizes. These tools have a limited impact on coding efficiency but a great one on complexity (up to a factor 60).

(ii) The use of Hadamard should be avoided since complexity is increased without any coding efficiency gain.

(iii) The adoption of multiple block sizes results in higher coding efficiency. While the complexity increases linearly with the number of block sizes, the major part of the gain is already achieved with the first 4 block sizes (16×6, 16×8, 8×16, and 8×8 pixels).

(iv) The CABAC entropy coder provides a consistent bit saving (10%) at the expense of a computational and memory increase (up to 30%) compared to UVLC.

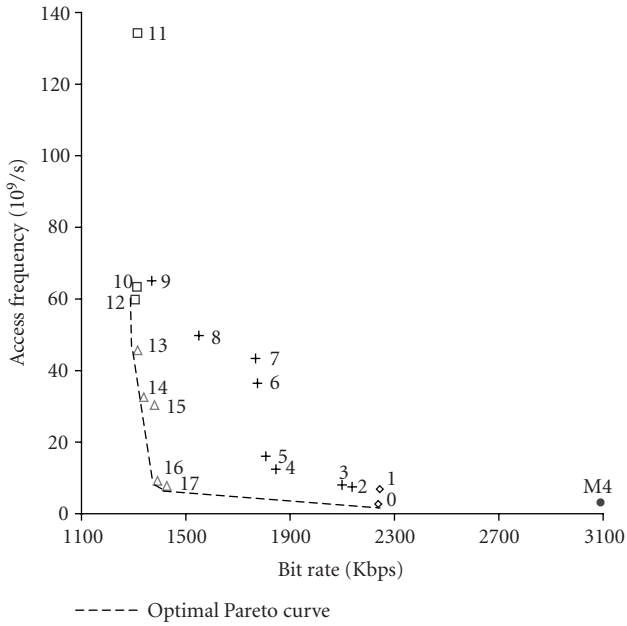


Figure 20: Encoder analysis, data transfer as cost in case of CM.

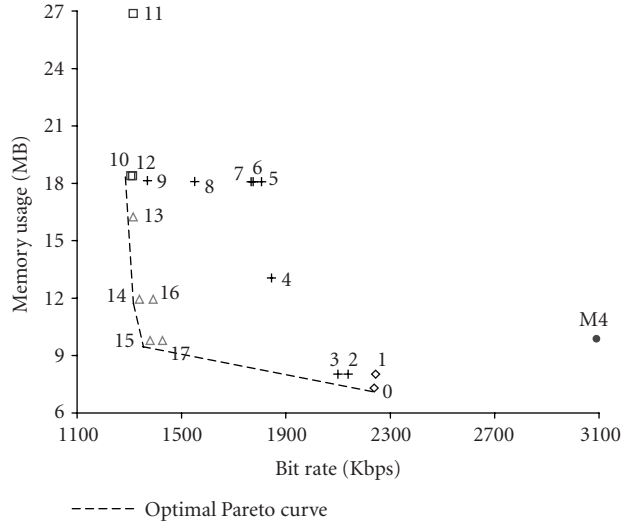


Figure 21: Encoder analysis, data storage as cost in case of CM.

(v) RD-Lagrangian techniques give a substantial compression efficiency improvement, but the complexity doubles when the codec configuration entails a lot of coding modes and motion estimation decisions.

(vi) For all video inputs, the use of B-frames reduces the bit rate (10% in average for the considered tests) for a complexity increase particularly noticeable at the decoder (from 20 to 40% extra cost specially at low rates). A larger number of B pictures increases the latency of the system (B frames need backward and forward reference pictures to be reconstructed). Thus, this tool is not used in low-latency constrained video applications and is typically not supported in baseline standard profiles [1, 5].

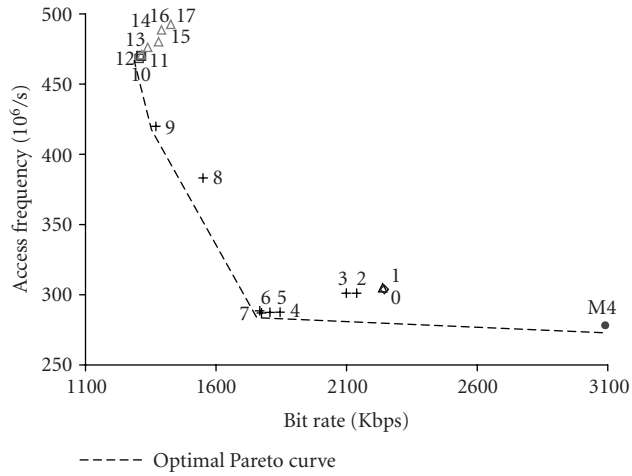


Figure 22: Decoder analysis, data transfer as cost in case of CM.

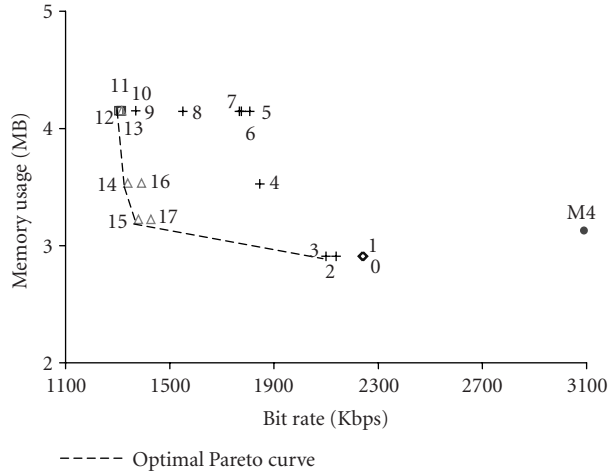


Figure 23: Decoder analysis, data storage as cost in case of CM.

The effect of some tools differs when applied to different sequences. Comparing the nonoptimal Pareto points in Figures 16, 17, 18, and 19 with their description in Table 2 provides useful hints on the AVC video tools for *video applications at low and middle bit rates (i.e., few tens up to hundreds of Kbps)*:

(i) the use of the eighth-pixel resolution leads to a complexity increase without any coding efficiency gain;

(ii) the use of B-frames for very low-bit-rate sequences as MD provides a low improvement in compression efficiency for the complexity increase it involves.

Different results emerge (see Figures 20, 21, 22, and 23 for the CM video test) when a similar analysis is applied to high-bit-rate video applications (thousands of Kbps):

(i) a higher pixel accuracy, eighth pixel instead of the basic quarter one, is a useful tool since it allows the same PSNR performance for at least 12% bit rate reduction (compare point 8 to point 7). The complexity increase is the same as for middle and low rates: roughly 15% for the encoder and 30% for the decoder as concerns data transfer and processing time. The impact on peak memory usage is negligible,

(ii) multiple reference frames are more useful (e.g., 5 reference frames lead to roughly 15% bit saving), where most of the bit saving is already achieved with 3 reference frames.

The above analysis is a static evaluation of the algorithmic performance and the required complexity to assess the efficiency of the video coding tools. It provides a basis for automatic tool selection and gives pointers for the development of a resource manager in future work.

7. AVC PROFILES

The results of the performance versus cost Pareto analysis in Section 6 provide inputs to assist the profile definition in the standard. A profile defines a set of coding tools that can be used for generating a conforming bitstream. All decoders conforming to a specific profile must support all features in that profile. Encoders are not required to make use of any particular set of features supported in a profile but they have to provide bitstreams decodable by conforming decoders. In AVC/H.264, three profiles are defined: the baseline, the extended, and the main profile [3]. With reference to the VCL video tools presented in Section 3.1,[2] the Baseline profile supports all new features in AVC (multireference frames, variable block sizes, quarter-pixel accuracy, in-loop deblocking, integer spatial transform, and spatial prediction for intracoding) except 1/8-pixel accuracy, CABAC and B pictures. The Extended profile supports all features of the Baseline profile plus B frames and some tools for error resiliency (e.g., switching pictures SP/SI and data partitioning [3]). The Main profile supports all VCL features described in Section 3.1 except eighth-pixel accuracy.[3] Baseline and Extended profiles are tailored for conversational services (typically operating below 1 Mbps) and streaming services (typically operating in the range 50–1500 Kbps), while entertainment video applications (several Mbps) would probably utilize the main profile.

The results of the VCL analysis presented in this paper are aligned with the profile considerations made by the standards body with the exception of the eighth-pixel accuracy which is no longer included in the last AVC release [3]. According to the results of Section 6, this choice is suitable for applications not targeting a high-rate, high-quality video scenario or when the low cost is the main issue (e.g., wireless video). In high-rate multimedia applications (e.g., thousands of Kbps for the test CM in Section 6), an increased pixel accuracy should be adopted since it leads to a noticeable coding efficiency gain. This consideration suggests that future extensions of the standard to high-quality video scenario, currently being considered by AVC, could/should envisage a pixel accuracy higher than quarter.

[2] A detailed analysis of AVC/H.264 profiles, including tools for error resiliency and network friendly data packaging, can be found in [3].

[3] The Main profile does not support some tools for error resiliency, such as flexible macroblock ordering [3], which are supported by the baseline profile. Thus, only a subset of the coded video sequences decodable by a baseline profile decoder can be decoded by a main profile decoder.

8. CONCLUSIONS

The advanced video codec (AVC) is recently defined in a joint standardization effort of ITU-T and ISO/IEC. This paper introduces this new video codec together with its performance and complexity co-evaluation. First, a description of the upcoming standard including both the encoder and the decoder architectures is addressed. Then, an exhaustive analysis of the coding efficiency versus complexity design space is carried out over a wide variety of video contents at the early algorithmic design phase. Since the increasing complexity of multimedia applications makes high-level system exploration time consuming and error pone, the co-evaluation approach is supported by a framework for automated analysis of the C-level specification. Different from known profiling methodologies, focusing mainly on PSNR, bit rate, and computational burden, the proposed approach also investigates memory metrics (data transfer and storage). Real-life implementations of H.263 and MPEG-4 systems demonstrate that multimedia applications are data dominated: data transfer and storage are the dominant cost factors for both hardware- and software-based architectures.

The simulation results show that AVC outperforms current video coding standards (up to 50% bit saving for the same PSNR) offering the enabling technology for a widespread diffusion of multimedia communication over wired and wireless transmission networks. However, this outstanding performance comes with an implementation complexity increase of a factor 2 for the decoder. At the encoder side, the cost increase is larger than one order of magnitude. This represents a design challenge for resource constrained multimedia systems such as wireless and/or wearable devices and high-volume consumer electronics, particularly for conversational applications (e.g., video telephony), where both the encoder and the decoder functionalities must be integrated in the user's terminal.

The analysis also highlights important properties of the AVC framework allowing for complexity reduction in the early algorithmic design phase. When combining the new coding features, the relevant implementation complexity accumulates, while the global compression efficiency saturates. As a consequence, a proper use of the AVC tools maintains roughly the same coding performance as the most complex configuration (all tools on) while considerably reducing complexity (up to a factor 6.5 for the encoder and 1.5 at the decoder side). A single AVC configuration able to maximize algorithmic performance while minimizing memory and computational burdens does not exist. However,

different configurations leading to several performance/cost trade-offs exist. To find these optimal configurations, and hence to highlight the bottlenecks of AVC, a Pareto multiobjective analysis is presented to explore the five-dimensional design space of PSNR, bit rate, computational burden, and memory access frequency and storage. The reported results provide inputs to assist the definition of profiles in the standard and represent the first step for a cost-effective implementation of the new AVC.

REFERENCES

[1] T. Wiegand, "Joint Final Committee Draft (JFCD) of joint video specification (ITU-T Rec. H.264, ISO/IEC 14496-10 AVC)," Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, JVT-D157, Klagenfurt, Austria, July 2002.

[2] ftp://ftp.imtc-files.org/jvt-experts/reference_software

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[4] D. Alfonso, D. Bagni, L. Celetto, and L. Pezzoni, "Detailed rate-distortion analysis of H.264 video coding standard and comparison to MPEG-2/4," in *Proc. of SPIE Conference on Visual Communications and Image Processing*, Lugano, Switzerland, July 2003.

[5] G. Côté, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849–866, 1998.

[6] P. Kuhn and W. Stechele, "Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation," in *Visual Communications and Image Processing*, vol. 3309 of *Proc. of SPIE*, pp. 498–509, San Jose, Calif, USA, January 1998.

[7] ISO/IEC 14496-2, "Information technology—Generic coding of audio-visual objects," N 3056, Maui, Hawaii, USA, December 1999.

[8] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, and A. Vandecapelle, *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design*, Kluwer Academic, Boston, Mass, USA, 1998.

[9] F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, and A. Vandecapelle, *Data Access and Storage Management for Embedded Programmable Processors*, Kluwer Academic, Boston, Mass, USA, 2002.

[10] J. Bormans, K. Denolf, S. Wuytack, L. Nachtergaele, and I. Bolsens, "Integrating system-level low power methodologies into a real-life design flow," in *IEEE Proc. 9th Int. Workshop Power and Timing Modeling Optimization and Simulation*, pp. 19–28, Kos, Greece, October 1999.

[11] A. Chimienti, L. Fanucci, R. Locatelli, and S. Saponara, "VLSI architecture for a low-power video codec system," *Microelectronics Journal*, vol. 33, no. 5-6, pp. 417–427, 2002.

[12] T. Meng, B. Gordon, E. Tsern, and A. Hung, "Portable video-on-demand in wireless communication," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 659–680, 1995.

[13] J. Jung, E. Lesellier, Y. Le Maguet, C. Miro, and J. Gobert, "Philips deblocking solution (PDS), a low complexity deblocking for JVT," Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, JVT-B037, Geneva, February 2002.

[14] L. Nachtergaele, D. Moolenaar, B. Vanhoof, F. Catthoor, and H. De Man, "System-level power optimization of video codecs on embedded cores: a systematic approach," *Journal of VLSI Signal Processing*, vol. 18, no. 2, pp. 89–111, 1998.

[15] K. Denolf, P. Vos, J. Bormans, and I. Bolsens, "Cost-efficient C-level design of an MPEG-4 video decoder," in *Proc. IEEE Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation: 10th International Workshop*, D. Soudris, P. Pirsch, and E. Barke, Eds., vol. 1918 of *Lecture Notes in Computer Science*, pp. 233–242, Springer-Verlag, Heidelberg, September 2000.

[16] M. Takahashi et al., "A 60-MHz 240-mW MPEG-4 videophone LSI with 16-Mb embedded DRAM," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1713–1721, 2000.

[17] http://www.imec.be/atomium.

[18] H.-J. Stolberg, M. Berekovic, and P. Pirsch, "A platform-independent methodology for performance estimation of streaming media applications," in *Proc. IEEE International Conference on Multimedia and Expo*, pp. 105–108, Lausanne, Switzerland, August 2002.

[19] C.-G. Zhou, L. Kohn, D. Rice, I. Kabir, A. Jabbi, and X.-P. Hu, "MPEG video decoding with the UltraSPARC visual instruction set," in *40th IEEE Computer Society International Conference*, pp. 470–475, San Francisco, Calif, USA, March 1995.

[20] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Francisco, Calif, USA, 1996.

[21] ISO/IEC 14496-5, "Information technology—Generic coding of audio-visual objects—Part 5: Amd1: Simulation software," N 3508, Beijing, July 2000.

[22] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 657–673, 2003.

[23] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.

[24] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[25] D. Marpe, G. Blättermann, G. Heising, and T. Wiegand, "Video compression using context-based adaptive arithmetic coding," in *Proc. IEEE International Conference on Image Processing*, vol. 3, pp. 558–561, Thessaloniki, Greece, October 2001.

[26] V. Lappalainen, A. Hallapuro, and T. Hämäläinen, "Optimization of emerging H.26L video encoder," in *Proc. IEEE Workshop on Signal Processing Systems*, pp. 406–415, Antwerp, Belgium, September 2001.

[27] V. Lappalainen, T. Hämäläinen, and A. Hallapuro, "Performance analysis of low bit rate H.26L video encoder," in *IEEE Trans. Acoustics, Speech, and Signal Processing*, pp. 1129–1132, May 2001.

[28] A. Joch, F. Kossentini, and P. Nasiopoulos, "A performance analysis of the ITU-T draft H.26L video coding standard," in *Proc. 12th International Packet Video Workshop*, Pittsburgh, Pa, USA, April 2002.

[29] M. Hagai, K. Abe, S. Kadono, S. Kajita, and M. Schlockerman, "H.26L implementation evaluation," ISO/IEC MPEG, document M7666, Pattaya, December 2001.

[30] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 704–716, 2003.

[31] K. Denolf et al., "Initial memory complexity analysis of the AVC codec," in *Proc. IEEE Workshop on Signal Processing Systems*, pp. 222–227, San Diego, Calif, USA, October 2002.

[32] H.-Y. Cheong and A. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc. IEEE International Conference on Multimedia and Expo*, pp. 517–520, Baltimore, Md, USA, July 2003.

[33] Y. Wang, S. Wenger, J. Wen, and A. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, 2000.
[34] V. Pareto, *Cours D'Economie Politique*, vol. I-II, Lausanne, Switzerland, 1896.
[35] E. Brockmeyer, A. Vandecappelle, and F. Catthoor, "Systematic cycle budget versus system power trade-off: a new perspective on system exploration of real-time data-dominated applications," in *Proc. International Symposium on Low Power Electronics and Design*, pp. 137–142, ACM Press, New York, 2000.

Sergio Saponara received his M.S. degree in electronics and his Ph.D. degree in information engineering, both from the University of Pisa, Italy, in 1999 and 2003. In 2001, he collaborated with "Consorzio Pisa Ricerche" on a MEDEA+ project related to the low-power design of xDSL modem. In 2002, he was with MICS group at IMEC, Leuven, Belgium, under a Marie Curie research scheme working on the complexity analysis of advanced video coding standards. Currently, he is a Researcher at Pisa University, working on algorithms and VLSI architecture design for multimedia and low-power CMOS design methodologies.

Kristof Denolf received the Industrial Engineering degree in electronics from the Katholieke Hogeschool, Brugge-Oostende, Belgium, in 1998, and the M.S. degree in electronic system design from Leeds Metropolitan University, Leeds, UK, in 2000. He joined the Multimedia Image Compression Systems (MICS) group at the Interuniversity Microelectronics Centre (IMEC), Leuven, Belgium, in 1998. His current research interests are the cost-efficient design of advanced video processing systems and the end-to-end quality of experience.

Gauthier Lafruit was a Research Scientist with the Belgian National Foundation for Scientific Research from 1989 to 1994, being mainly active in the area of wavelet image compression implementations. Subsequently, he was a Research Assistant in the Vrije Universiteit Brussel (VUB), Belgium. In 1996, he became the recipient of the Scientific Barco Award and joined IMEC, where he was involved as a Senior Scientist with the design of low-power VLSI for combined JPEG/wavelet compression engines. He is currently the Principal Scientist in the MICS group with IMEC. His main interests include progressive transmission in still image, video and 3D object coding, as well as scalability and resource monitoring for advanced, scalable video, and 3D coding applications. He is the author/coauthor of around sixty scientific publications, fifty MPEG standardization contributions, and five patent(s) (applications) and has participated (and has been appointed as Evaluator) in several national and international project(s) (proposals).

Carolina Blanch was born in Pamplona, Spain. She received her M.S. degree in telecommunications engineering in January 2000 from the University of Navarra, Spain. She received a scholarship from the Flemish community to continue her studies in Belgium, and obtained an M.S. in artificial intelligence from the Katholieke Universiteit Leuven in 2001. Since December 2001, she is working as a Research Engineer in the MICS group at IMEC, Leuven, Belgium. She has been involved in the complexity assessment of video codecs and has several related articles. Currently, she is involved in cross layer optimization and network adaptation of video codecs.

Jan Bormans has been a Researcher at the Information Retrieval and Interpretation Sciences laboratory, the Vrije Universiteit Brussel (VUB), Belgium, in 1992 and 1993. In 1994, he joined the VLSI Systems and Design Methodologies (VSDM) division of the IMEC research center in Leuven, Belgium. Since 1996, he is heading IMEC's Multimedia Image Compression Systems group. This group focuses on the efficient design and implementation of embedded systems for advanced multimedia applications. He is the Belgian head of the delegation for ISO/IEC's MPEG and SC29 standardization committees. He is also MPEG-21 requirements Editor and Chairman of the MPEG liaison group.

EURASIP Journal on Applied Signal Processing 2004:2, 236–252

New Complexity Scalable MPEG Encoding Techniques for Mobile Applications

Stephan Mietens
Philips Research Laboratories, Prof. Holstlaan 4, NL-5656 AA Eindhoven, The Netherlands
Email: stephan.mielens@philips.com

Peter H. N. de With
LogicaCMG Eindhoven, Eindhoven University of Technology, P.O. Box 7089, Luchthavenweg 57, NL-5600 MB Eindhoven, The Netherlands
Email: p.h.n.de.with@tue.nl

Christian Hentschel
Cottbus University of Technology, Universitätsplatz 3-4, D-03044 Cottbus, Germany
Email: christian.hentschel@tu-cottbus.de

Received 10 December 2002; Revised 7 July 2003

Complexity scalability offers the advantage of one-time design of video applications for a large product family, including mobile devices, without the need of redesigning the applications on the algorithmic level to meet the requirements of the different products. In this paper, we present complexity scalable MPEG encoding having core modules with modifications for scalability. The interdependencies of the scalable modules and the system performance are evaluated. Experimental results show scalability giving a smooth change in complexity and corresponding video quality. Scalability is basically achieved by varying the number of computed DCT coefficients and the number of evaluated motion vectors, but other modules are designed such they scale with the previous parameters. In the experiments using the "Stefan" sequence, the elapsed execution time of the scalable encoder, reflecting the computational complexity, can be gradually reduced to roughly 50% of its original execution time. The video quality scales between 20 dB and 48 dB PSNR with unity quantizer setting, and between 21.5 dB and 38.5 dB PSNR for different sequences targeting 1500 kbps. The implemented encoder and the scalability techniques can be successfully applied in mobile systems based on MPEG video compression.

Keywords and phrases: MPEG encoding, scalable algorithms, resource scalability.

1. INTRODUCTION

Nowadays, digital video applications based on MPEG video compression (e.g., Internet-based video conferencing) are popular and can be found in a plurality of consumer products. While in the past, mainly TV and PC systems were used, having sufficient computing resources available to execute the video applications, video is increasingly integrated into devices such as portable TV and mobile consumer terminals (see Figure 1).

Video applications that run on these products are heavily constrained in many aspects due to their limited resources as compared to high-end computer systems or high-end consumer devices. For example, real-time execution has to be assured while having limited computing power and memory for intermediate results. Different video resolutions have to be handled due to the variable displaying of video frame sizes. The available memory access or transmission bandwidth is limited as the operating time is shorter for computation-intensive applications. Finally the product success on the market highly depends on the product cost. Due to these restrictions, video applications are mainly redesigned for each product, resulting in higher production cost and longer time-to-market.

In this paper, it is our objective to design a scalable MPEG encoding system, featuring scalable video quality and a corresponding scalable resource usage [1]. Such a system enables advanced video encoding applications on a plurality of low-cost or mobile consumer terminals, having limited resources (available memory, computing power, stand-by time, etc.) as compared to high-end computer systems or high-end consumer devices. Note that the advantage of scalable systems is that they are designed once for a whole product family instead of a single product, thus they have a faster

Figure 1: Multimedia applications shown on different devices sharing the available resources.

time-to-market. State-of-the-art MPEG algorithms do not provide scalability, thereby hampering, for example, low-cost solutions for portable devices and varying coding applications in multitasking environments.

This paper is organized as follows. Section 2 gives a brief overview of the conventional MPEG encoder architecture. Section 3 gives an overview of the potential scalability of computational complexity in MPEG core functions. Section 4 presents a scalable discrete cosine transformation (DCT) and motion estimation (ME), which are the core functions of MPEG coding systems. Part of this work was presented earlier. A special section between DCT and ME is devoted to content-adaptive processing, which is of benefit for both core functions. The enhancements on the system level are presented in Section 5. The integration of several individual scalable functions into a full scalable coder has given a new framework for experiments. Section 6 concludes the paper.

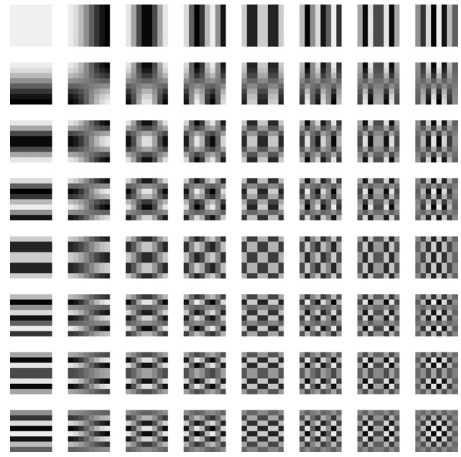


Figure 2: DCT block of basis patterns.

2. CONVENTIONAL MPEG ARCHITECTURE

The MPEG coding standard is used to compress a video sequence by exploiting the spatial and temporal correlations of the sequence as briefly described below.

Spatial correlation is found when looking into individual video frames (pictures) and considering areas of similar data structures (color, texture). The DCT is used to decorrelate spatial information by converting picture blocks to the transform domain. The result of the DCT is a block of transform coefficients, which are related to the frequencies contained in the input picture block. The patterns shown in Figure 2 are the representation of the frequencies, and each picture block is a linear combination of these basis patterns. Since high frequencies (at the bottom right of the figure) commonly have lower amplitudes than other frequencies and are less perceptible in pictures, they can be removed by quantizing the DCT coefficients.

Temporal correlation is found between successive frames of a video sequence when considering that the objects and background are on similar positions. For data compression purpose, the correlation is removed by predicting the contents and coding the frame differences instead of complete frames, thereby saving bandwidth and/or storage space. Motion in video sequences introduced by camera movements or moving objects result in high spatial frequencies occurring in the frame difference signal. A high compression rate is achieved by predicting picture contents using ME and motion compensation (MC) techniques.

For each frame, the above-mentioned correlations are exploited differently. Three different types of frames are defined in the MPEG coding standard, namely, I-, P-, and B-frames. I-frames are coded as completely independent frames, thus only spatial correlations are exploited. For P- and B-frames, temporal correlations are exploited, where P-frames use one temporal reference, namely, the past reference frame. B-frames use both the past and the upcoming reference frames, where I-frames and P-frames serve as reference frames. After MC, the frame difference signals are coded by DCT coding.

A conventional MPEG architecture is depicted in Figure 3. Since B-frames refer to future reference frames, they cannot be encoder/decoder before this reference frame is received by the coder (encoder or decoder). Therefore, the video frames are processed in a reordered way, for example, "IPBB" (transmit order) instead of "IBBP" (display order).

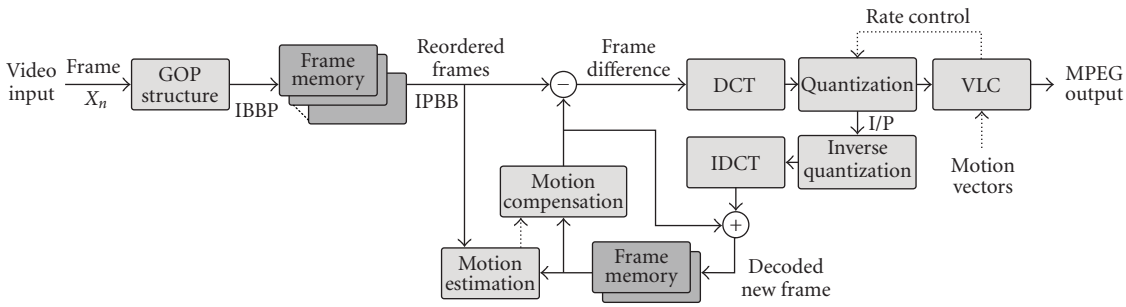


Figure 3: Basic architecture of an MPEG encoder.

Note that for the ME process, reference frames that are used are reduced in quality due to the quantization step. This limits the accuracy of the ME. We will exploit this property in the scalable ME.

3. SCALABILITY OVERVIEW OF MPEG FUNCTIONS

Our first step towards scalable MPEG encoding is to redesign the individual MPEG core functions (modules) and make them scalable themselves. In this paper, we concentrate mainly on scalability techniques on the algorithmic level, because these techniques can be applied to various sorts of hardware architectures. After the selection of an architecture, further optimizations on the core functions can be made. An example to exploit features of a reduced instruction set computer (RISC) processor for obtaining an efficient implementation of an MPEG coder is given in [2].

In the following, the scalability potentials of the modules shown in Figure 3 are described. Further enhancements that can be made by exploiting the modules interconnections are described in Section 5. Note that we concentrate on the encoder and do not consider pre- or postprocessing steps of the video signal, because such steps can be performed independently from the encoding process. For this reason, the input video sequence is modified neither in resolution nor in frame rate for achieving reduced complexity.

GOP structure

This module defines the types of the input frames to form group of pictures (GOP) structures. The structure can be either *fixed* (all GOPs have the same structure) or *dynamic* (content-dependent definition of frame types). The computational complexity required to define fixed GOP structures is negligible. Defining a dynamic GOP structure has a higher computational complexity, for example for analyzing frame contents. The analysis is used for example to detect scene changes. The rate distortion ratio can be optimized if a GOP starts with the frame following the scene change.

Both the fixed and the dynamic definitions of the GOP structure can control the computational complexity of the coding process and the bit rate of the coded MPEG stream with the ratio of I-, P-, and B-frames in the stream. In general, I-frames require less computation than P- or B-frames, because no ME and MC is involved in the processing of I-frames. The ME, which requires significant computational effort, is performed for each temporal reference that is used. For this reason, P-frames (having one temporal reference) are normally half as complex in terms of computations as B-frames (having two temporal references). It can be considered further that no inverse DCT and quantization is required for B-frames. For the bit rate, the relation is the other way around since each temporal reference generally reduces the amount of information (frame contents or changes) that has to be coded.

The chosen GOP structure has influence on the memory consumption of the encoder as well, because frames must be kept in memory until a reference frame (I- or P-frame) is processed. Besides defining I-, P-, and B-frames, input frames can be skipped and thus are not further processed while saving memory, computations, and bit rates.

The named options are not further worked out, because they can be easily applied on every MPEG encoder without the need to change the encoder modules themselves. A dynamic GOP structure would require additional functionality through, for example, scene change detection. The experiments that are made for this paper are based on a fixed GOP structure.

Discrete cosine transformation

The DCT transforms image blocks to the transform domain to obtain a powerful compression. In conjunction with the inverse DCT (IDCT), a perfect reconstruction of the image blocks is achieved while spending fewer bits for coding the blocks than not using the transformation. The accuracy of the DCT computation can be lowered by reducing the number of bits that is used for intermediate results. In principle, reduced accuracy can scale up the computation speed because several operations can be executed in parallel (e.g., two 8-bit operations instead of one 16-bit operation). Furthermore, the silicon area needed in hardware design is scaled down with reduced accuracy due to simpler hardware components (e.g., an 8-bit adder instead of a 16-bit adder). These two possibilities are not further worked out because they are not algorithm-specific optimizations and therefore are suitable for only a few hardware architectures.

An algorithm-specific optimization that can be applied on any hardware architecture is to scale down the number of DCT coefficients that are computed. A new technique, considering the baseline DCT algorithm and a corresponding architecture for finding a specific computation order of the coefficients, is described in Section 4.1. The computation order maximizes the number of computed coefficients for a given limited amount of computation resources.

Another approach for scalable DCT computation predicts at several stages during the computation whether a group of DCT coefficients are zero after quantization and their computation can be stopped or not [3].

Inverse discrete cosine transformation

The IDCT transforms the DCT coefficients back to the spatial domain in order to reconstruct the reference frames for the (ME) and (MC) process. The previous discussion on scalability options for the DCT also applies to the IDCT. However, it should be noted that a scaled IDCT should have the same result as a perfect IDCT in order to be compatible with the MPEG standard. Otherwise, the decoder (at the receiver side) should ensure that it uses the same scaled IDCT as in the encoder in order to avoid error drift in the decoded video sequence.

Previous work on scalability of the IDCT at the receiver side exists [4, 5], where a simple subset of the received DCT coefficients is decoded. This has not been elaborated because in this paper, we concentrate on the encoder side.

Quantization

The quantization reduces the accuracy of the DCT coefficients and is therefore able to remove or weight frequencies of lower importance for achieving a higher compression ratio. Compared to the DCT where data dependencies during the computation of 64 coefficients are exploited, the quantization processes single coefficients where intermediate results cannot be reused for the computation of other coefficients. Nevertheless, computing the quantization involves rounding that can be simplified or left out for scaling up the processing speed. This possibility has not been worked out further.

Instead, we exploit scalability for the quantization based on the scaled DCT by preselecting coefficients for the computation such that coefficients that are not computed by the DCT are not further processed.

Inverse quantization

The inverse quantization restores the quantized coefficient values to the regular amplitude range prior to computing the IDCT. Like the IDCT, the inverse quantization requires sufficient accuracy to be compatible with the MPEG standard. Otherwise, the decoder at the receiver should ensure that it avoids error drift.

Motion estimation

The ME computes motion vector (MV) fields to indicate block displacements in a video sequence. A picture block (macroblock) is then coded with reference to a block in a previously decoded frame (the prediction) and the difference to this prediction. The ME contains several scalability options. In principle, any good state-of-the-art fast ME algorithm offers an important step in creating a scaled algorithm. Compared to full search, the computing complexity is much lower (significantly less MV candidates are evaluated) while accepting some loss in the frame prediction quality. Taking the fast ME algorithms as references, a further increase of the processing speed is obtained by simplifying the applied set of motion vectors (MVs).

Besides reducing the number of vector candidates, the displacement error measurement (usually the sum of absolute pixel differences (SAD)) can be simplified (thus increase computation speed) by reducing the number of pixel values (e.g., via subsampling) that are used to compute the SAD. Furthermore, the accuracy of the SAD computation can be reduced to be able to execute more than one operation in parallel. As described for the DCT, this technique is suitable for a few hardware architectures only.

Up to this point, we have assumed that ME is performed for each macroblock. However, the number of processed macroblocks can be reduced also, similar to the pixel count for the SAD computation. MVs for omitted macroblocks are then approximated from neighboring macroblocks. This technique can be used for concentrating the computing effort on areas in a frame, where the block contents lead to a better estimation of the motion when spending more computing power [6].

A new technique to perform the ME in three stages by exploiting the opportunities of high-quality frame-by-frame ME is presented in Section 4.3. In this technique, we used several of the above-mentioned options and we deviate from the conventional MPEG processing order.

Motion compensation

The MC uses the MV fields from the ME and generates the frame prediction. The difference between this prediction and the original input frame is then forwarded to the DCT. Like the IDCT and the inverse quantization, the MC requires sufficient accuracy for satisfying the MPEG standard. Otherwise, the decoder (at the receiver) should ensure using the same scaled MC as in the encoder to avoid error drift.

Variable-length coding (VLC)

The VLC generates the coded video stream as defined in the MPEG standard. Optimization of the output can be made here, like ensuring a predefined bit rate. The computational effort is scalable with the number of nonzero coefficients that remain after quantization.

4. SCALABLE FUNCTIONS FOR MPEG ENCODING

Computationally expensive corner stones of an MPEG encoder are the DCT and the ME. Both are addressed in the scalable form in Section 4.1 on the scalable DCT [7] and in Section 4.3 on the scalable ME [8], respectively. Additionally,

Section 4.2 presents a scalable block classification algorithm, which is designed to support and integrate the scalable DCT and ME on the system level (see Section 5).

4.1. Discrete Cosine Transformation

4.1.1. Basics

The DCT transforms the luminance and chrominance values of small square blocks of an image to the transform domain. Afterwards, all coefficients are quantized and coded. For a given $N \times N$ image block represented as a two-dimensional (2D) data matrix $\{X[i, j]\}$, where $i, j = 0, 1, \ldots, N-1$, the 2D DCT matrix of the coefficients $\{Y[m, n]\}$ with $m, n = 0, 1, \ldots, N-1$ is computed by

$$Y[m,n] = \frac{4}{N^2} * u(m) * u(n) * \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X[i,j] * \cos\frac{(2i+1)m * \pi}{2N} * \cos\frac{(2j+1)n * \pi}{2N}, \quad (1)$$

where $u(i) = 1/\sqrt{2}$ if $i = 0$ and $u(i) = 1$ elsewhere. Equation (1) can be simplified by ignoring the constant factors for convenience and defining a square cosine matrix K by

$$K_N[p,q] = \cos\frac{(2p+1)q * \pi}{2N} \quad (2)$$

so that (1) can be rewritten as

$$Y = K_N * X * K_N^{\top}. \quad (3)$$

Equation (3) shows that the 2D DCT as specified by (1) is based on two orthogonal 1D DCTs, where $K_N * X$ transforms the columns of the image block X, and $X * K_N^{\top}$ transforms the rows. Since the computation of two 1D DCTs is less expensive than one 2D DCT, state-of-the-art DCT algorithms normally refer to (3) and concentrate on optimizing a 1D DCT.

4.1.2. Scalability

Our proposed scalable DCT is a novel technique for finding a specific computation order of the DCT coefficients. The results depend on the applied (fast) DCT algorithm. In our approach, the DCT algorithm is modified by eliminating several computations and thus coefficients, thereby enabling complexity scalability for the used algorithm. Consequently, the output of the algorithm will have less quality, but the processing effort of the algorithm is reduced, leading to a higher computing speed. The key issue is to identify the computation steps that can be omitted to maximize the number of coefficients for the best possible video quality.

Since fast DCT algorithms process video data in different ways, the algorithm used for a certain scalable application should be analyzed closely as follows. Prior to each computation step, a list of remaining DCT coefficients is sorted

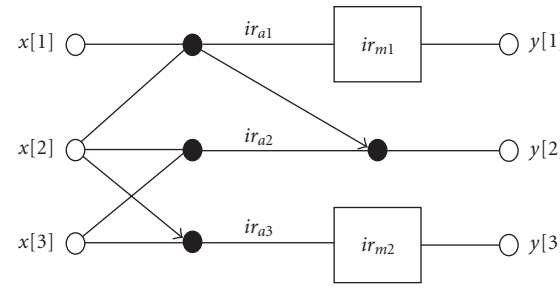


FIGURE 4: Exemplary butterfly structure for the computation of outputs $y[\cdot]$ based on inputs $x[\cdot]$. The data flow of DCT algorithms can be visualized using such butterfly diagrams.

such that in the next step, the coefficient is computed having the lowest computational cost. More formally, the sorted list $L = \{l_1, l_2, \ldots, l_{N^2}\}$ of coefficients l taken from an $N \times N$ DCT satisfies the condition

$$C(l_i) = \min_{k \geq i} C(l_k), \quad \forall l_i \in L \quad (4)$$

where $C(l_k)$ is a cost function providing the remaining number of operations required for the coefficient l_k given the fact that the coefficients l_n, $n < k$, already have been computed. The underlying idea is that some results of previously performed computations can be shared. Thus (4) defines a minimum computational effort needed to obtain the next coefficient.

We give a short example of how the computation order L is obtained. In Figure 4, a computation with six operation nodes is shown, where three nodes are intermediate results (ir_{a1}, ir_{a2}, and ir_{a3}). The complexity of the operations that are involved for a node can be defined such that they represent the characteristics (like CPU usage or memory access costs) of the target architecture. For this example, we assume that the nodes depicted with filled circles (•) require one operation and nodes that are depicted with squares (□) require three operations. Then, the outputs (coefficients) $y[1]$, $y[2]$, and $y[3]$ require 4, 3, and 4 operations, respectively. In this case, the first coefficient in list L is $l_1 = y[2]$ because it requires the least number of operations. Considering that, with $y[2]$, the shared node ir_1 has been computed and its intermediate result is available, the remaining coefficients $y[1]$ and $y[3]$ require 3 and 4 operations, respectively. Therefore, $l_2 = y[1]$ and $l_3 = y[3]$, leading to a computation order $L = \{y[2], y[1], y[3]\}$.

The computation order L can be perceptually optimized if the subsequent quantization step is considered. The quantizer weighting function emphasizes the use of low-frequency coefficients in the upper-left corner of the matrix. Therefore, the cost function $C(l_k)$ can be combined with a priority function to prefer those coefficients.

Note that the computation order L is determined by the algorithm and the optional applied priority function, and it can be found in advance. For this reason, no computational

	0	1	2	3	4	5	6	7
0	1	33	9	41	5	44	14	36
1	17	49	21	57	29	63	31	55
2	10	37	3	42	11	39	7	48
3	25	61	26	53	18	51	24	60
4	6	45	15	34	2	35	16	46
5	28	59	23	52	19	54	27	62
6	12	47	8	40	13	43	4	38
7	20	56	32	64	30	58	22	50

Figure 5: Computation order of coefficients.

overhead is required for actually computing the scaled DCT. It is possible, though, to apply different precomputed DCTs to different blocks employing block classification that indicates which precomputed DCT should perform best with a classified block (see Section 5.3).

4.1.3. Experiments

For experiments, the fast 2D algorithm given by Cho and Lee [9], in combination with the Arai-Agui-Nakajima (AAN) 1D algorithm [10], has been used, and this algorithm combination is extended in the following with computational complexity scalability. Both algorithms were adopted because their combination provides a highly efficient DCT computation (104 multiplications and 466 additions). The results of this experiment presented below are discussed with the assumption that an addition is equal to one operation and a multiplication is equal to three operations (in powerful cores, additions and multiplications have equal weight).

The scalability-optimized computation order in this experiment is shown in Figure 5, where the matrix has been shaded with different gray levels to mark the first and the second half of the coefficients in the sorted list. It can be seen that in this case, the computation order clearly favors horizontal or vertical edges (depending on whether the matrix is transposed or not).

Figure 6 shows the scalability of our DCT computation technique using the scalability-optimized computation order, and the zigzag order as reference computation order. In Figure 6a, it can be seen that the number of coefficients that are computed with the scalability-optimized computation order is higher at any computation limit than the zigzag order. Figure 6b shows the resulting peak signal-to-noise ratio (PSNR) of the first frame from the "Voit" sequence using both computation orders, where no quantization step is performed. A 1–5 dB improvement in PSNR can be noticed, depending on the amount of available operations.

Finally, Figure 7 shows two picture pairs (based on zigzag and scalability-optimized orders preferring horizontal details) sampled from the "Renata" sequence during different stages of the computation (representing low-cost and medium-cost applications). Perceptive evaluations of our experiments have revealed that the quality improvement of our technique is the largest between 200 and 600 operations per block. In this area, the amount of coefficients is still relatively small so that the benefit of having much more coefficients computed than in a zigzag order is fully exploited. Although the zigzag order yields perceptually important coefficients from the beginning, the computed number is simply too low to show relevant details (e.g., see the background calendar in the figure).

4.2. Scalable classification of picture blocks

4.2.1. Basics

The conventional MPEG encoding system processes each image block in the same content-independent way. However, content-dependent processing can be used to optimize the coding process and output quality, as indicated below.

(i) Block classification is used for quantization to distinguish between flat, textured, and mixed blocks [11] and then apply different quantization factors for these blocks for optimizing the picture quality at given bit rate limitations. For example, quantization errors in textured blocks have a small impact on the perceived image quality. Blocks containing both flat and textured parts (mixed blocks) are usually blocks that contain an edge, where the disturbing ringing effect gets worse with high quantization factors.

(ii) The ME (see Section 4.3) can take the advantage of classifying blocks to indicate whether a block has a structured content or not. The drawback of conventional ME algorithms that do not take the advantage of block classification is that they spend many computations on computing MVs for, for example, relatively flat blocks. Unfortunately, despite the effort, such ME processes yield MVs of poor quality. Employing block classification, computations can be concentrated on blocks that may lead to accurate MVs [12].

Of course, in order to be useful, the costs to perform block classification should be less than the saved computations. Given the above considerations, in the following, we will adopt content-dependent adaptivity for coding and motion processing. The next section explains the content adaptivity in more detail.

4.2.2. Scalability

We perform a simple block classification based on detecting horizontal and vertical transitions (edges) for two reasons.

(i) From the scalable DCT, computation orders are available that prefer coefficients representing horizontal or vertical edges. In combination with a classification, the computation order that fits best for the block content can be chosen.

(ii) The ME can be provided with the information whether it is more likely to find a good MV in up-down or left-right search directions. Since ME will find equally

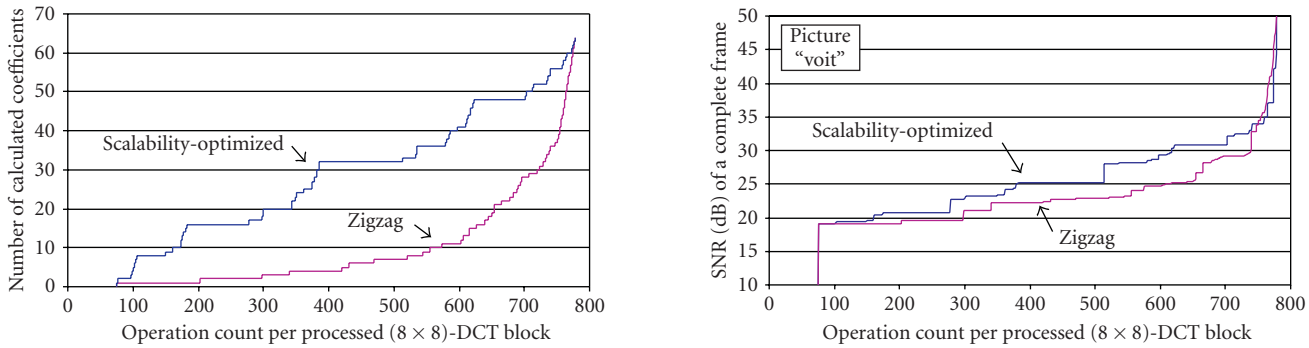


Figure 6: Comparison of the scalability-optimized computation order with the zigzag order. At limited computation resources, more DCT coefficients are computed (a) and a higher PSNR is gained (b) with the scalability-optimized order than with the zigzag order.

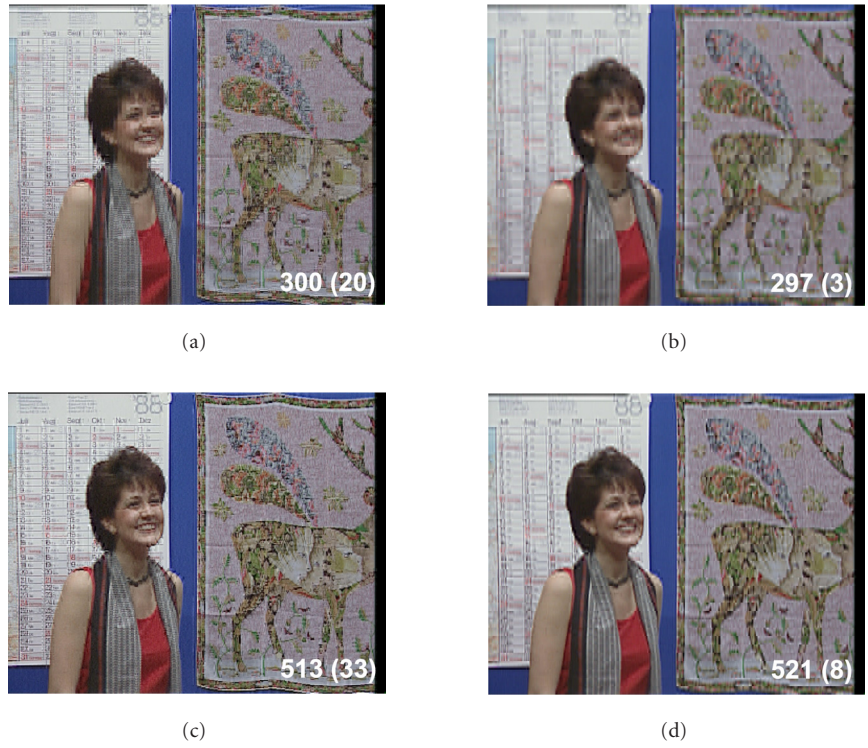
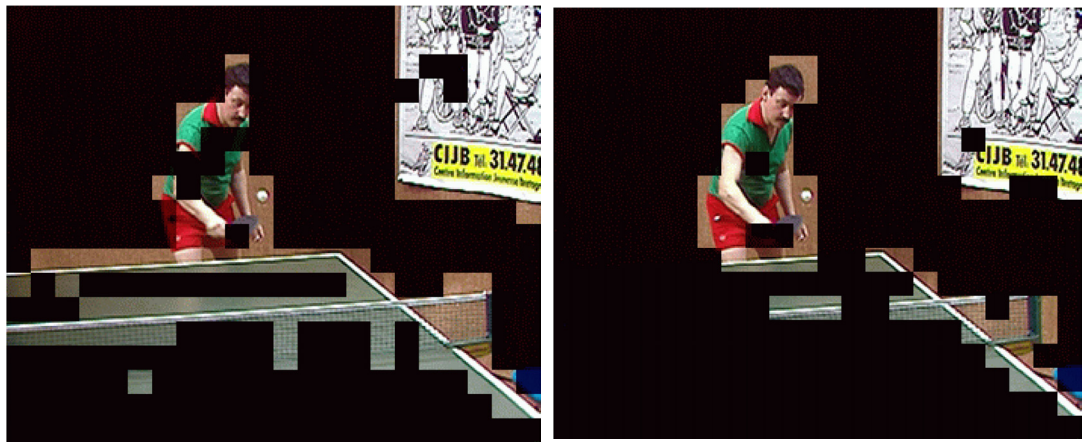


Figure 7: A video frame from the "Renata" sequence coded employing the scalability-optimized order (a) and (c), and the zigzag order (b) and (d). Index $m(n)$ means m operations are performed for n coefficients. The scalability-optimized computation order results in an improved quality (compare sharpness and readability).

good MVs for every position *along* such an edge (where a displacement in this direction does not introduce large displacement errors), searching for MVs *across* this edge will rapidly reduce the displacement error and thus lead to an appropriate MV. Horizontal and vertical edges can be detected by significant changes of pixel values in vertical and horizontal directions, respectively.

The edge detecting algorithm we use is in principle based on continuously summing up pixel differences along rows or columns and counting how often the sum exceeds a certain threshold. Let p_i, with $i = 0, 1, \ldots, 15$, be the pixel values in a row or column of a macroblock (size 16×16). We then define a range where pixel divergence (d_i) is considered as noise if $|d_i|$ is below a threshold t. The pixel divergence is defined by Table 1.



(a) (b)

Figure 8: Visualization of block classification using a picture of the "table tennis" sequence. The left (right) picture shows blocks where horizontal (vertical) edges are detected. Blocks that are visible in both pictures belong to the class "diagonal/structured," while blocks that are blanked out in both pictures are considered as "flat."

Table 1: Definition of pixel divergence, where the divergence is considered as noise if it is below a certain threshold.

Condition	Pixel divergence d_i
$i = 0$	0
$(i = 1,\ldots,15) \wedge (\lvert d_{i-1}\rvert \leq t)$	$d_{i-1} + (p_i - p_{i-1})$
$(i = 1,\ldots,15) \wedge (\lvert d_{i-1}\rvert > t)$	$d_{i-1} + (p_i - p_{i-1}) - \operatorname{sgn}(d_{i-1}) * t$

The area preceding the edge yields a level in the interval $[-t;+t]$. The middle of this interval is at $d = 0$, which is modified by adding $\pm t$ in the case that $|d|$ exceeds the interval around zero (start of the edge). This mechanism will follow the edges and prevent noise from being counted as edges. The counter c as defined below indicates how often the actual interval was exceeded:

$$c = \sum_{i=1}^{15} \begin{cases} 0 & \text{if } |d_i| \leq t, \\ 1 & \text{if } |d_i| > t. \end{cases} \tag{5}$$

The occurrence of an edge is defined by the resulting value of c from (5).

This edge detecting algorithm is scalable by selecting the threshold t, the number of rows and columns that are considered for the classification, and a typical value for c. Experimental evidence has shown that in spite of the complexity scalability of this classification algorithm, the evaluation of a single row or column in the middle of a picture block was found sufficient for a rather good classification.

4.2.3. Experiments

Figure 8 shows the result of an example to classify image blocks of size 16×16 pixels (macroblock size). For this experiment, a threshold of $t = 25$ was used. We considered a block to be classified as a "horizontal edge" if $c \geq 2$ holds for the central column computation and as a "vertical edge" if $c \geq 2$ holds for the row computation. Obviously, we can derive two extra classes: "flat" (for all blocks that do not belong to the CLASS "horizontal edge" NOR the class "vertical edge") and diagonal/structured (for blocks that belong to both classes horizontal edge and vertical edge).

The visual results of Figure 8 are just an example of a more elaborate set of sequences with which experiments were conducted. The results showed clearly that the algorithm is sufficiently capable of classifying the blocks for further content-adaptive processing.

4.3. Motion estimation

4.3.1. Basics

The ME process in MPEG systems divides each frame into rectangular macroblocks (16×16 pixels each) and computes MVs per block. An MV signifies the displacement of the block (in the x-y pixel plane) with respect to a reference image. For each block, a number of candidate MVs are examined. For each candidate, the block evaluated in the current image is compared with the corresponding block fetched from the reference image displaced by the MV. After testing all candidates, the one with the best match is selected. This match is done on basis of the SAD between the current block and the displaced block. The collection of MVs for a frame forms an MV field.

State-of-the-art ME algorithms [13, 14, 15] normally concentrate on reducing the number of vector candidates for a single-sided ME between two frames, independent of the frame distance. The problem of these algorithms is that a higher frame distance hampers accurate ME.

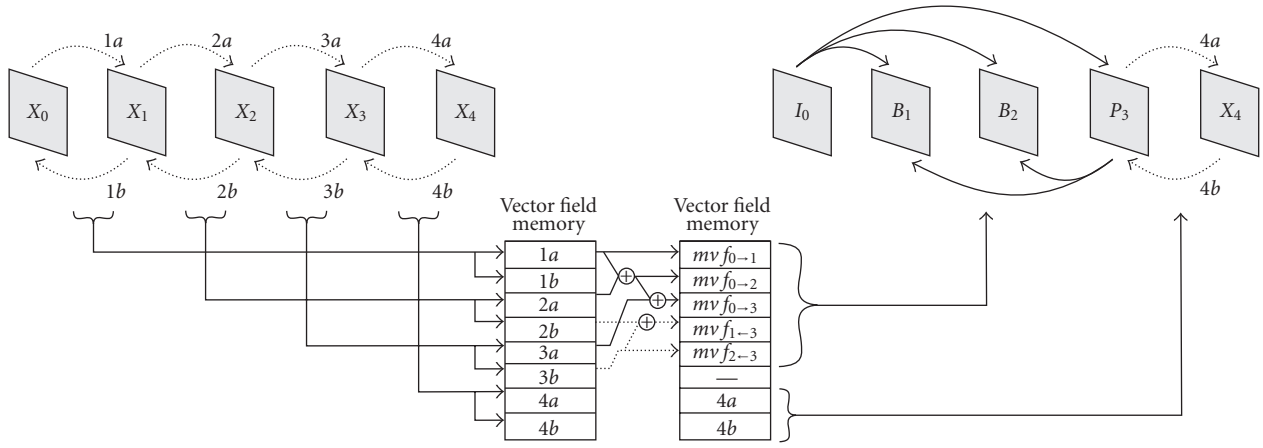


Figure 9: An overview of the new scalable ME process. Vector fields are computed for successive frames (left) and stored in memory. After defining the GOP structure, an approximation is computed (middle) for the vector fields needed for MPEG coding (right). Note that for this example it is assumed that the approximations are performed after the exemplary GOP structure is defined (which enables dynamic GOP structures), therefore the vector field $(1b)$ is computed but not used afterwards. With predefined GOP structures, the computation of $(1b)$ is not necessary.

4.3.2. Scalability

The scalable ME is designed such that it takes the advantage of the intrinsically high prediction quality of ME between successive frames (smallest temporal distance), and thereby works not only for the typical (predetermined and fixed) MPEG GOP structures, but also for more general cases. This feature enables on-the-fly selection of GOP structures depending on the video content (e.g., detected scene changes, significant changes of motion, etc.). Furthermore, we introduce a new technique for generating MV fields from other vector fields by *multitemporal* approximation (not to be confused with other forms of multitemporal ME as found in H.264). These new techniques give more flexibility for a scalable MPEG encoding process.

The estimation process is split up into three stages as follows.

Stage 1 Prior to defining a GOP structure, we perform a simple recursive motion estimation (RME) [16] for every received frame to compute the forward and backward MV field between the received frame and its predecessor (see the left-hand side of Figure 9). The computation of MV fields can be omitted for reducing computational effort and memory.

Stage 2 After defining a GOP structure, all the vector fields required for MPEG encoding are generated through multitemporal approximations by summing up vector fields from the previous stage. Examples are given in the middle of Figure 9, for example, vector field $(mvf_{0\to3}) = (1a) + (2a) + (3a)$. Assume that the vector field $(2a)$ has not been computed in Stage 1 (due to a chosen scalability setting), one possibility to approximate $(mvf_{0\to3})$ is $(mvf_{0\to3}) = 2 * (1a) + (3a)$.

Stage 3 For final MPEG ME in the encoder, the computed approximated vector fields from the previous stage are used as an input. Beforehand, an optional refinement of the approximations can be performed with a second iteration of simple RME.

We have employed simple RME as a basis for introducing scalability because it offers a good quality for time-consecutive frames at low computing complexity.

The presented three-stage ME algorithm differs from known multistep ME algorithms like in [17], where initially estimated MPEG vector fields are processed for a second time. Firstly, we do not have to deal with an increasing temporal distance when deriving MV fields in Stage 1. Secondly, we process the vector fields in a display order having the advantage of frame-by-frame ME, and thirdly, our algorithm provides scalability. The possibility of scaling vector fields, which is part of our multitemporal predictions, is mentioned in [17] but not further exploited. Our algorithm makes explicit use of this feature, which is a fourth difference. In the sequel, we explain important system aspects of our algorithm.

Figure 10 shows the architecture of the three-stage ME algorithm embedded in an MPEG encoder. With this architecture, the initial ME process in Stage 1 results in a high-quality prediction because original frames without quantization errors are used. The computed MV fields can be used in Stage 2 to optimize the GOP structures. The optional refinement of the vector fields in Stage 3 is intended for high-quality applications to reach the quality of a conventional MPEG ME algorithm.

The main advantage of the proposed architecture is that it enables a broad scalability range of resource usage and achievable picture quality in the MPEG encoding process. Note that a bidirectional ME (usage of B-frames) can be realized at the same cost of a single-directional ME (usage of P-frames only) when properly scaling the computational

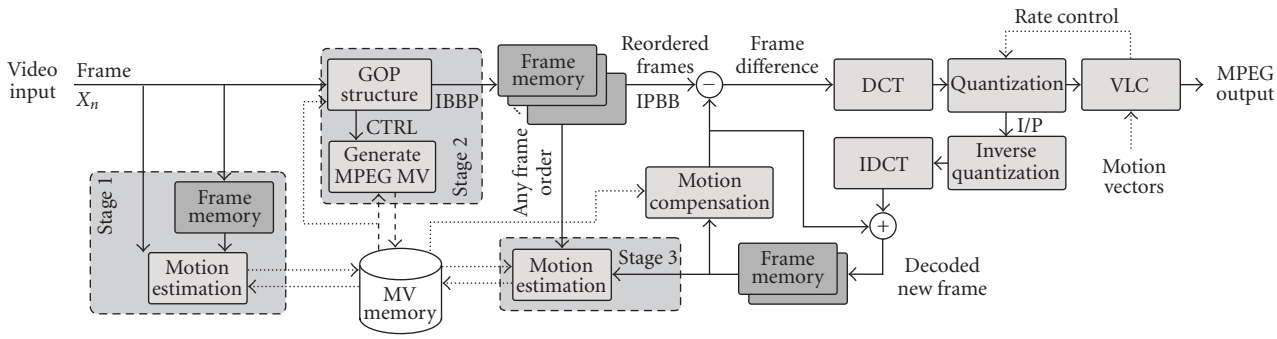


Figure 10: Architecture of an MPEG encoder with the new scalable three-stage motion estimation.

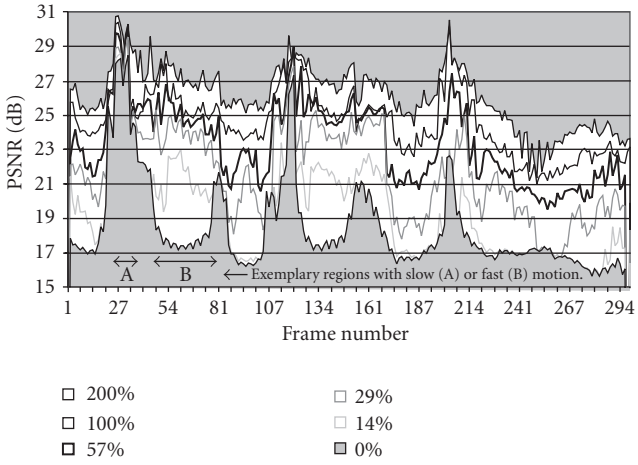


Figure 11: PSNR of motion-compensated B-frames of the "Stefan" sequence (tennis scene) at different computational efforts—P-frames are not shown for the sake of clarity ($N = 16$, $M = 4$). The percentage shows the different computational effort that results from omitting the computation of vector fields in Stage 1 or performing an additional refinement in Stage 3.

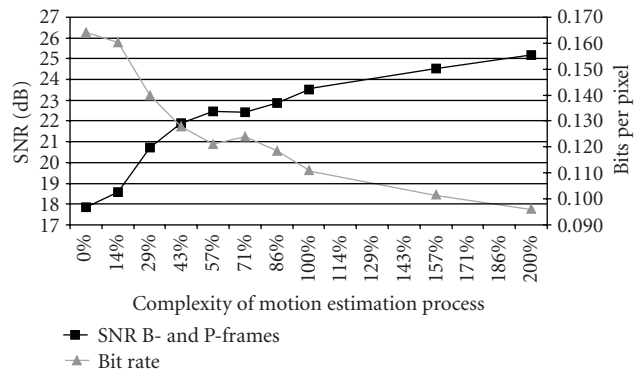


Figure 12: Average PSNR of motion-compensated P- and B-frames and the resulting bit rate of the encoded "Stefan" stream at different computational efforts. A lower average PSNR results in a higher differential signal that must be coded, which leads to a higher bit rate. The percentage shows the different computational effort that results from omitting the computation of vector fields in Stage 1 or performing an additional refinement in Stage 3.

complexity, which makes it affordable for mobile devices that up till now rarely make use of B-frames. A further optimization is seen (but not worked out) in limiting the ME process of Stages 1 and 3 to significant parts of a vector field in order to further reduce the computational effort and memory.

4.3.3. Experiments

To demonstrate the flexibility and scalability of the three-stage ME technique, we conducted an initial experiment using the "Stefan" sequence (tennis scene). A GOP size of $N = 16$ and $M = 4$ (thus "IBBBP" structure) was used, combined with a simple pixel-based search. In this experiment, the scaling of the computational complexity is introduced by gradually increasing the vector field computations in Stage 1 and Stage 3. The results of this experiment are shown in Figure 11. The area in the figure with the white background shows the scalability of the quality range that results from downscaling the amount of computed MV fields. Each vector field requires 14% of the effort compared to a 100% simple RME [16] based on four forward vector fields and three backward vector fields when going from one to the next reference frame. If all vector fields are computed and the refinement Stage 3 is performed, the computational effort is 200% (not optimized).

The average PSNR of the motion-compensated P- and B-frames (taken after MC and before computing the differential signal) of this experiment and the resulting bit rate of the encoded MPEG stream are shown in Figure 12. Note that for comparison purpose, no bit rate control is performed during encoding and therefore, the output quality of the MPEG streams for all complexity levels is equal. The quantization factors, *qscale*, we have used are 12 for I-frames and 8 for P- and B-frames. For a full quality comparison (200%), we consider a full-search block matching with a search window of 32×32 pixels. The new ME technique slightly outperforms this full search by 0.36 dB PSNR measured from the motion-compensated P- and B-frames of this experiment (25.16 dB instead of 24.80 dB). The bit rate of the complete MPEG

Table 2: Average luminance PSNR of the motion-compensated P- and B-frames for sequences "Stefan" (A), "Renata" (B), and "Teeny" (C) with different ME algorithms. The second column shows the average number of SAD-based vector evaluations per MV (based on (A)).

Algorithm	Tests/MV	(A)	(B)	(C)
2D FS (32×32)	926.2	24.80	29.62	26.78
NTSS [14]	25.2	22.55	27.41	24.22
Diamond [15]	21.9	22.46	27.34	26.10
Simple RME [16]	16.0	21.46	27.08	23.89
Three-stage ME 200% (employing [16])	37.1	25.16	29.24	26.92
Three-stage ME 100% (employing [16])	20.1	23.52	27.45	24.74

sequence is 0.012 bits per pixel (bpp) lower when using the new technique (0.096 bpp instead of 0.108 bpp). When reducing the computational effort to 57% of a single-pass simple RME, an increase of the bit rate by 0.013 bpp compared to the 32×32 full search (FS) is observed.

Further comparisons are made with the scalable three-stage ME running at full and "normal" quality. Table 2 shows the average PSNR of the motion-compensated P- and B-frames for three different video sequences and ME algorithms with the same conditions as described above (same N, M, etc.). The first data column (tests per MV) shows the average number of vector tests that are performed per macroblock in the "Stefan" sequence to indicate the performance of the algorithms. Note that MV tests pointing outside the picture are not counted, which results in numbers that are lower than the nominal values (e.g., 926.2 instead of 1024 for 32×32 FS). The simple RME algorithm results in the lowest quality here because only three vector field computations out of $4 * (4 + 3) = 28$ can use temporal vector candidates as prediction. However, our new three-stage ME that uses this simple RME performs, comparable to FS, at 200% complexity, and at 100%, it is comparable to the other fast ME algorithms.

The results in Table 2 are based on the simple RME algorithm from [16]. A modified algorithm has been found later [18] that forms an improved replacement for the simple RME. This modified algorithm is based on the block classification as presented in Section 4.2. This algorithm was used for further experiments and is summarized as follows. Prior to estimating the motion between two frames, the macroblocks inside a frame are classified into areas having horizontal, vertical edges, or no edges. The classification is exploited to minimize the number of MV evaluations for each macroblock by, for example, concentrating vector evaluations across the detected edge. A novelty in the algorithm is a *distribution* of good MVs *to* other macroblocks, even already processed ones, which differs from other known recursive ME techniques that reuse MVs *from previously* processed blocks.

5. SYSTEM ENHANCEMENTS AND EXPERIMENTS

The key approach to optimize a system is to reuse and combine data that is generated by the system modules in order to control other modules. In the following, we present several approaches, where data can be reused or generated at a low cost in a coding system for an optimization purpose.

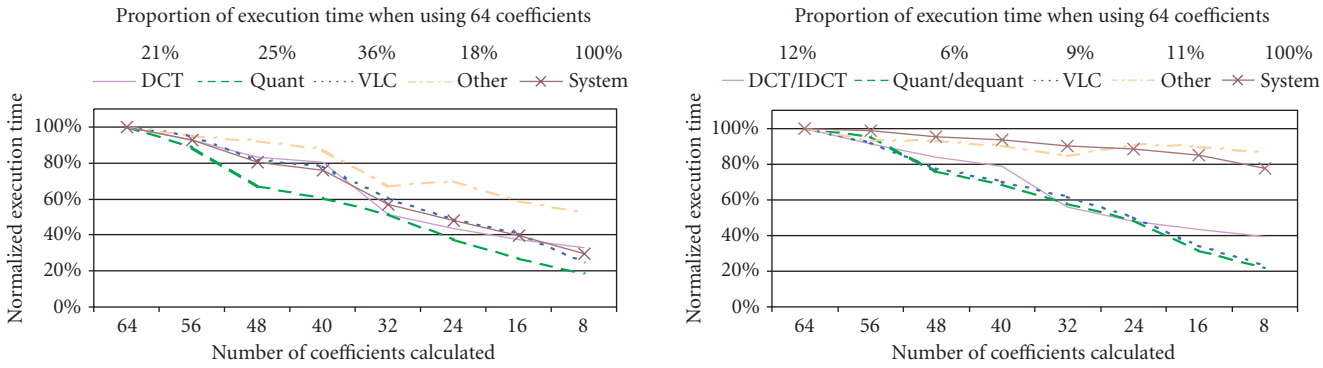
5.1. Experimental environment

The scalable modules for the (I)DCT, (de)quantization, ME, and VLC are integrated into an MPEG encoder framework, where the scaling of the IDCT and the (de)quantization is effected from the scalable DCT (see Section 5.2). In order to visualize the obtained scalability of the computations, the scalable modules are executed at different parameter settings, leading to effectively varying the number of DCT coefficients and MV candidates evaluated. When evaluating the system complexity, the two different numbers have to be combined into a joint measure. In the following, the elapsed execution time of the encoder needed to code a video sequence is used as a basis for comparison. Although this time parameter highly depends on the underlying architecture and on the programming and operating system, it reflects the complexity of the system due to the high amount of operations involved.

The experiments were conducted on a Pentium-III Linux system running at 733 MHz. In order to be able to measure the execution time of single functions being part of the complete encoder execution, it was necessary to compile the C++ program of the encoder without compiler optimizations. Additionally, it should be noted that the experimental C++ code was not optimized for fast execution or usage of architecture-specific instructions (e.g., MMX). For these reasons, the encoder and its measured execution times cannot be compared with existing software-based MPEG encoders. However, we have ensured that the measured change in the execution time results from the scalability of the modules, as we did not change the programming style, code structures, or common coding parameters.

5.2. Effect of scalable DCT

The fact that a scaled DCT computes only a subset S of all possible DCT coefficients C can be used for the optimization of other modules. The subset S is known before the subsequent quantization, dequantization, VLC, and IDCT modules. Of course, coefficients that are not computed are set to zero and therefore they do not have to be processed further in any of these modules. Note that because the subset S is known in advance, no additional tests are performed to



(a) (1,1)-GOP (I-frames only).

(b) (12,4)-GOP (IBBBP structure).

Figure 13: Complexity reduction of the encoder modules relative to the full DCT processing, with (1,1)-GOPs (a) and with (12,4)-GOPs) (b). Note that in this case, 62% of the coding time is spent in (b) for ME and MC (not shown for convenience). For visualization of the complexity reduction, we normalize the execution time for each module to 100% for full processing.

detect zero coefficients. This saves computations as follows.

(i) The quantization and dequantization require a fixed amount of operations per processed intra- or intercoefficient. Thus, each skipped coefficient $c \in C \setminus S$ saves 1/64 of the total complexity of the quantization and dequantization modules.

(ii) The VLC processes the DCT coefficients in a zigzag or an alternate order and generates run-value pairs for coefficients that are unequal to zero. "Run" indicates the number of zero coefficients that are skipped before reaching a nonzero coefficient. The usage of a scaled DCT increases the probability that zero coefficients occur, for which no computations are spent.

(iii) The IDCT can be simplified by knowing which coefficients are zero. It is obvious that, for example, each multiplication with a known factor of 0 and additions with a known addend of 0 can be skipped.

The execution time of the modules when coding the "Stefan" sequence and scaling the modules that process coefficients is visualized in Figure 13. The category "other" is used for functions that are not exclusively used by the scaled modules. Figure 13a shows the results of an experiment, where the sequence was coded with I-frames only. Similar results are observed in Figure 13b from another experiment, for which P- and B-frames are included. To remove the effect of quantization, the experiments were performed with qscale = 1. In this way, the figures show results that are less dependent on the coded video content.

The measured PSNR of the scalable encoder running at full quality is 46.5 dB for Figure 13a and 48.16 dB for Figure 13b. When the number of computed coefficients is gradually reduced from 64 to 8, the PSNR drops gradually to 21.4 dB Figure 13a, respectively, 21.81 dB in Figure 13b. In Figures 13a and 13b, the quality gradually reduces from "no noticeable differences" down to "severe blockiness." In Figure 13b, the curve for the ME module is not shown for convenient because the ME (in this experiment, we used diamond search ME [15]) is not affected from processing a different number of DCT coefficients.

5.3. Selective DCT computation based on block classification

The block classification introduced in Section 4.2 is used to enhance the output quality of the scaled DCT by using different computation orders for blocks in different classes. A simple experiment indicates the benefit in quality improvement. In the experiment, we computed the average values of DCT coefficients when coding the "table tennis" sequence with I-frames only. Each DCT block is taken after quantization with qscale = 1. Figure 14 shows the statistic for blocks that are classified as having a horizontal (left graph) or vertical (right graph) edge only. It can be seen that the classification leads to a frequency concentration in the DCT coefficient matrix in the first column, respectively, row.

We found that the DCT algorithm of Arai et al. [10] can be used best for blocks with horizontal or vertical edges, while background blocks have a better quality impression when using the algorithm by Cho and Lee [9]. The experiment made for Figure 15 shows the effect of the two algorithms on the table edges ([10] is better) and the background ([9] is better). In both cases, the computation orders designed for preferring horizontal edges are used. The computation limit was set to 256 operations, leading to 9 computed coefficients for [10] and 11 for [9], respectively. The coefficients that are computed are marked in the corresponding DCT matrix. It can be seen that [10] covers all main vertical frequencies, while [9] covers a mixture of high and low vertical and horizontal frequencies. The resulting overall PSNR are 26.58 dB and 24.32 dB, respectively.

Figure 16 shows the effect of adaptive DCT computation based on classification. Almost all of the background blocks were classified as flat blocks and therefore, ChoLee was chosen for these blocks. For convenient, both algorithms were set

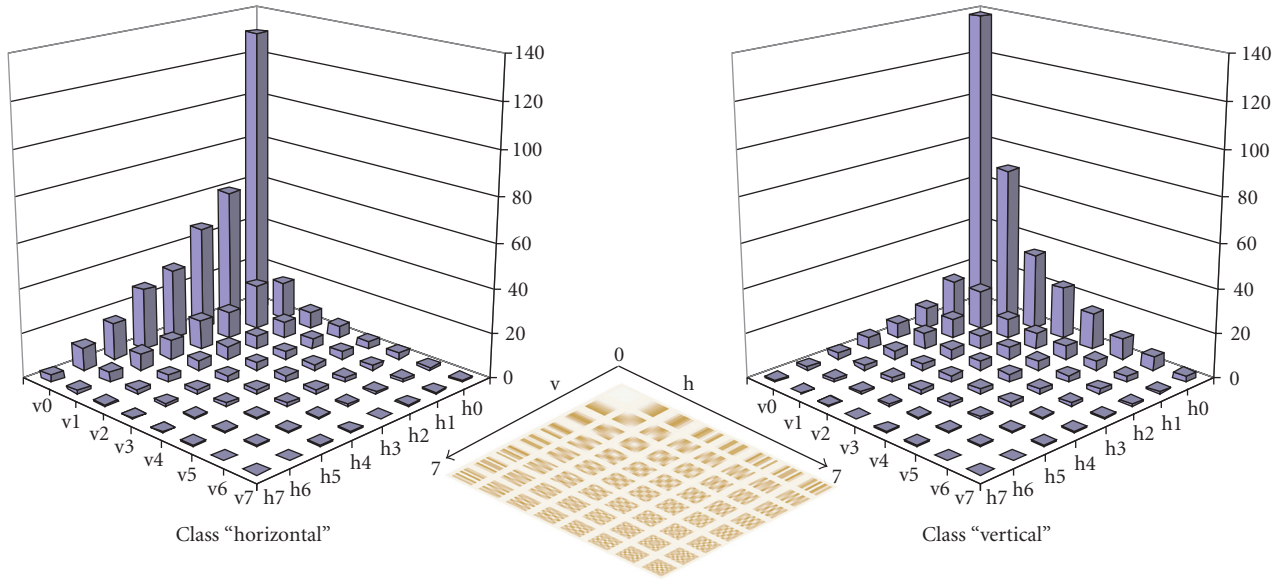


FIGURE 14: Statistics of the average absolute values of the DCT coefficients taken after quantization with qscale = 1. Here, the "table tennis" sequence was coded with I-frames only. The left (right) graph shows the statistic for blocks classified as having horizontal (vertical) edges.

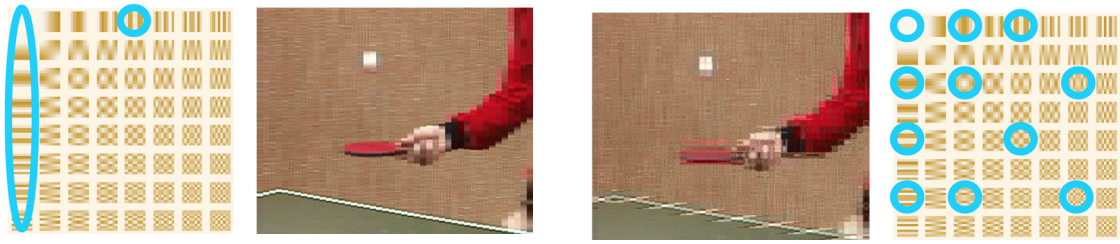


FIGURE 15: Example of scaled AAN-DCT (a) and ChoLee-DCT (b) at 256 operations. AAN fits better for horizontal edges, while ChoLee has better results for the background.

to compute 11 coefficients. Blocks with both detected horizontal and vertical edges are treated as blocks having horizontal edges only because an optimized computation order for such blocks is not yet defined. The resulting PSNR is 26.91 dB.

5.4. Dynamic interframe DCT coding

Besides intraframe coding, the DCT computation on frame differences (for interframe coding) occurs more often than intraframe coding ($N - 1$ times for (N, M) GOPs). For this reason, we look more closely to interframe DCT coding, where we discovered a special phenomenon from the scalable DCT. It was found that the DCT coded frame differences show temporal fluctuations in frequency content. The temporal fluctuation is caused by the motion in the video content combined with the special selection function of the coefficients computed in our scalable DCT. Due to the motion, the energy in the coefficients shifts over the selection pattern so that the quality gradually increases over time. Figure 17 shows this effect from an experiment when coding the "Stefan" sequence with IPP frames (GOP structure (GOP size N, IP distance M) = $(12, 1)$) while limiting the computation to 32 coefficients. The camera movement in the shown sequence is panning to the right. It can be seen for example that the artifacts around text decrease over time.

The aforementioned phenomenon was mainly found in sequences containing not too much motion. The described effect leads to the idea of temporal data partitioning using a cyclical sequence of several scalable DCTs with different coefficient selection functions. The complete cycle would compute each coefficient at least once. Temporal data partitioning means that the computational complexity of the DCT computation is spread over time, thereby reducing the average complexity of the DCT computation (per block) at the expense of obtaining delayed quality obtainment. Using this technique, picture blocks having a static content (blocks

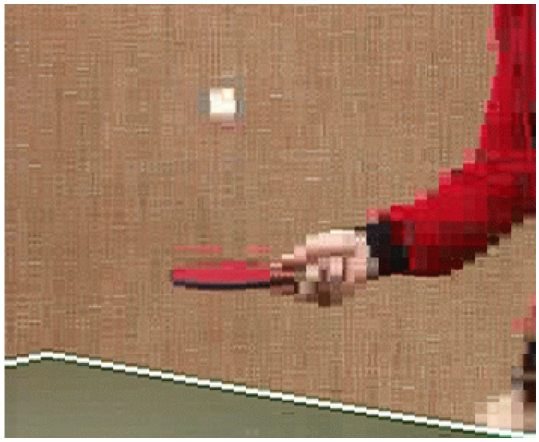


Figure 16: Both DCT algorithms were used to code this frame. After block classification, the ChoLee-DCT was used to code blocks where no edges were detected and the AAN-DCT for blocks with detected edges.

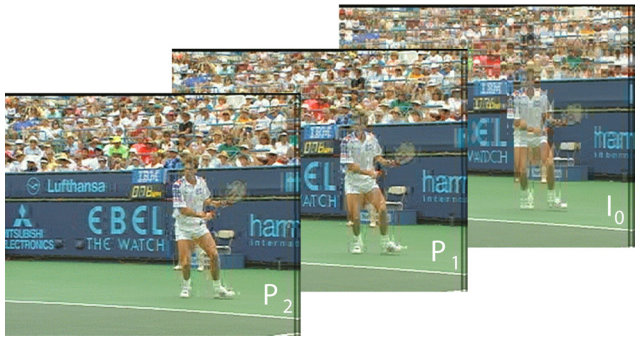


Figure 17: Visualization of a phenomenon from the scalable DCT, leading to a gradual quality increase over time.

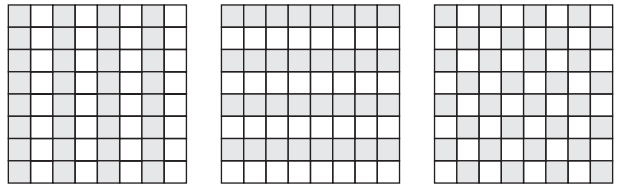


Figure 18: Example of coefficient subsets (marked gray) used for dynamic interframe DCT coding with a limitation to 32 coefficients per subset.

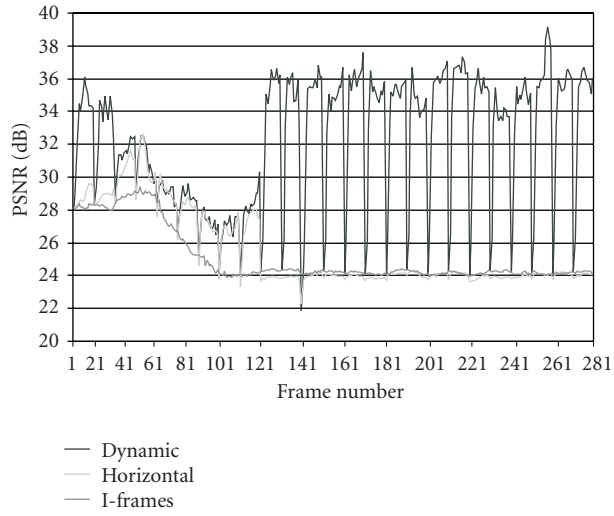


Figure 19: PSNR measures for the coded "table tennis" sequence, where the DCT computation was scaled to compute 32 coefficients. Compared to coding I-frames only (medium gray curve), inter DCT coding results in an improved output quality in case of motion (light gray curve) and even a higher output quality with dynamic interframe DCT computation.

having zero motion like nonmoving background) and therefore having no temporal fluctuations in their frequency content will obtain the same result as a nonpartitioned DCT computation after full computation of the partitioned DCT.

Based on the idea of temporal data partitioning, we define N subsets s_i (with $i = 0, \ldots, N-1$) of coefficients such that

$$\bigcup_{i=0}^{N-1} s_i = S, \tag{6}$$

where the set S contains all the 64 DCT coefficients. The subsets s_i are used to build up functions f_i that compute a scaled DCT for the coefficients in s_i. The functions f_i are applied to blocks with static contents in cyclical sequence (one per intercoded frame). After N intercoded frames, each coefficient for these blocks is computed at least once.

We set up an experiment using the "table tennis" sequence as follows in order to measure the effect of dynamic interframe coding. The computation of the DCT (for intraframe coding and interframe coding) was limited to 32 coefficients. The coefficient subsets we used are shown in Figure 18. Figure 19 shows the improvement in the PSNR that is achieved with this approach. Three curves are shown in this figure, plotting the achieved PSNR of the coded frames. The medium gray curve results from coding all the frames as I-frames, which we take as a reference in this experiment. The other two curves result from applying a GOP structure with $N = 16$ and $M = 4$. First, all blocks are processed with a fixed DCT (light gray curve) computing only the coefficients as shown in the left subset of Figure 18. It can be seen that when the content of the sequence changes due to movement, the PSNR increases. Second, the dynamic inter-DCT coding technique is applied to the coding process, which results in the dark gray curve. The dark gray curve shows an improvement to the light gray curve in case of no motion. The comb-like structure of the curve results from the periodic I-frame occurrence that restarts the quality buildup. The low periodicity of the quality drop gives a visually annoying effect that can be solved by computing more

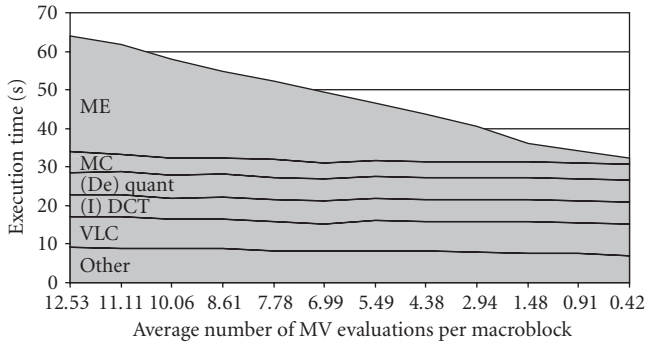


Figure 20: Example of ME scalability for the complete encoder when using a (12, 4)-GOP ("IBBBP" structure) for coding.

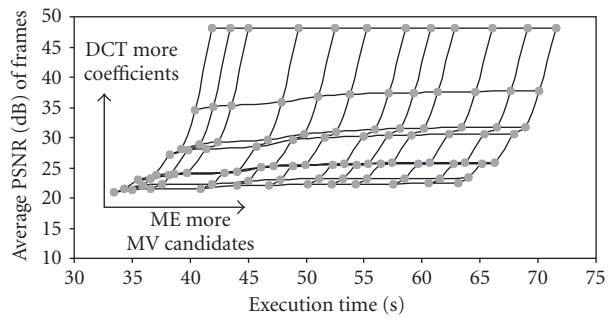


Figure 21: PSNR results of different configurations for the scalable MPEG modules.

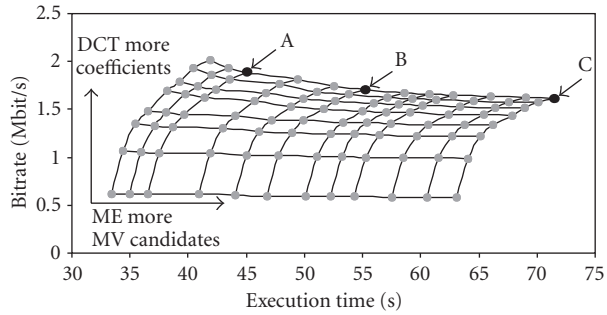


Figure 22: Obtained bit rates of different configurations for the scalable modules. The markers refer to points in the design space, where the same bit rate and quality (not computational complexity) is obtained as resulting from using diamond search (A) or full search with a 32×32 (B) or 64×64 (C) search area for ME.

coefficients for the I-frames. Although this seems interesting, this was not further pursued because of limited time.

5.5. Effect of scalable ME

The execution time of the MPEG modules when coding the "Stefan" sequence and scaling the ME is visualized in Figure 20. It can be seen that the curve for the ME block scales linearly with the number of MV evaluations, whereas the other processing blocks remain constant. The average number of vector candidates that are evaluated per macroblock by the scalable ME in this experiment is between 0.42 and 12.53. This number is clearly below the achieved average number of candidates (21.77) when using the diamond search [15]. At the same time, we found that our scalable codec results in a higher quality of the MC frame (up to 25.22 dB PSNR in average) than the diamond search (22.53 dB PSNR in average), which enables higher compression ratios (see the next section).

5.6. Combined effect of scalable DCT and scalable ME

In this section, we combine the scalable ME and DCT in the MPEG encoder and apply the scalability rules for (de)quantization, IDCT, and VLC, as we have described them in Section 2. Since the DCT and ME are the main sources for scalability, we will focus on the tradeoff between MVs and the number of computed coefficients.

Figure 21 portrays the obtained average PSNR of the coded "Stefan" sequence (CIF resolution) and Figure 22 shows the achieved bit rate corresponding to Figure 21. The experiments are performed with a (12,4)-GOP and qscale = 1. Both figures indicate the large design space that is available with the scalable encoder without quantization and open-loop control. The horizontally oriented curves refer to a fixed number of DCT coefficients (e.g., $8, 16, 24, 32, \ldots, 64$), whereas vertically oriented curves refer to a fixed number of MV candidates. A normal codec would compute all the 64 coefficients and would therefore operate on the top horizontal curve of the graph. The figures should be jointly evaluated. Under the above-mentioned measurement conditions, the potential benefit of the scalable ME is only visible in the reduction of the bit rate (see Figure 22) since an improved ME leads to less DCT coefficients for coding the difference signal after the MC in the MPEG loop.

In Figure 22, it can be seen that the bit rate decreases when computing more MV candidates (going to the right). The reduction is only visible when the bit rate is high enough. For comparison, the markers "A," "B", and "C" refer to three points from the design space. With these markers, the obtained bit rate of the scalable encoder is compared to the encoder using another ME algorithm. Marker "A" refers to the configuration of the encoder using the scalable ME, where the same bit rate and video quality (not the computational complexity) are achieved compared to the diamond search. As mentioned earlier, the diamond search performs 21.77 MV candidates on the average per macroblock. Our scalable coder operating under the same quality and bit rate combination as the diamond search in marker "A" results in 10.06 average MV candidates, thus 53.8% less than the diamond search. Markers "B" and "C" result from using the full-search ME with a 32×32 and 64×64 search area, respectively, requiring substantially more vector candidates (1024 and 4096, respectively). Figure 21 shows a corresponding measurement with the average PSNR, as the outcome, instead of the bit rate.

Figures 21 and 22 both present a large design space, but in practice, this is limited due to the quantization and bit rate control. Further experiments using quantization and bit rate control at 1500 kbps for the "Stefan," "Foreman," and "table tennis" sequence resulted in a quality level range from roughly 22 dB to 38 dB. As could be expected from inserting the quantization, the curves moved to lower PSNR (the lower half of Figure 21) and less computation time is required since fewer coefficients are computed. It was found that the remaining design space is larger for sequences having less motion.

6. CONCLUSIONS

We have presented techniques for complexity scalable MPEG encoding that gradually reduce the quality as a function of limited resources. The techniques involve modifications to the encoder modules in order to pursue scalable complexity and/or quality. Special attention has been paid to exploiting a scalable DCT and ME because they represent two computational expensive corner stones of MPEG encoding. The introduced new techniques for the scalability of the two functions show considerable savings of computational complexity for video applications having low-quality requirements. In addition, a scalable block classification technique has been presented, which is designed to support the scalable processing of the DCT and ME. In the second step, performance evaluations have been carried out by constructing a complete MPEG encoding system in order to show the design space that is achieved with the scalability techniques. It has been shown that even a higher reduction in computational complexity of the system could be obtained if available data (e.g., which DCT coefficients are computed during a scalable DCT computation) is exploited to optimize other core functions.

The obtained execution times of the encoder when coding the "Stefan" sequence as an example for complexity has been measured. It was found that the overall execution time of the scalable encoder can be gradually reduced to roughly 50% of its original execution time. At the same time, the codec provides a wide range of video quality levels (roughly from 20 dB to 48 dB PSNR in average) and compression ratios (from 0.58 to 2.02 Mbps). Further experiments targeting a bit rate of 1500 kbps for the Stefan, Foreman, and table tennis sequence result in a quality level range from roughly 21.5 dB to 38.5 dB. Compared with the diamond search ME from literature which requires 21.77 MV candidates on the average per macroblock, our scalable coder operating under the same quality and bit rate combination uses 10.06 average MV candidates, thus 53.8% less than the diamond search.

Another result of our experiments is that the scalable DCT has an integrated coefficient selection function which may enable a quality increase during interframe coding. This phenomenon can lead to an MPEG encoder with a number of special DCTs with different selection functions, and this option should be considered for future work. This should also include different scaling of the DCT for intra- and interframe coding. For scalable ME, future work should examine the scalability potentials of using various fixed and dynamic GOP structures, and of concentrating or limiting the ME to frame parts, whose content (could) have the current viewer focus.

REFERENCES

[1] C. Hentschel, R. Braspenning, and M. Gabrani, "Scalable algorithms for media processing," in *IEEE International Conference on Image Processing (ICIP '01)*, vol. 3, pp. 342–345, Thessaloniki, Greece, October 2001.

[2] R. Prasad and K. Ramkishor, "Efficient implementation of MPEG-4 video encoder on RISC core," in *IEEE International Conference on Consumer Electronics, Digest of Technical papers (ICCE '02)*, pp. 278–279, Los Angeles, Calif, USA, June 2002.

[3] K. Lengwehasatit and A. Ortega, "DCT computation based on variable complexity fast approximations," in *Proc. IEEE International Conference of Image Processing (ICIP '98)*, vol. 3, pp. 95–99, Chicago, Ill, USA, October 1998.

[4] S. Peng, "Complexity scalable video decoding via IDCT data pruning," in *International Conference on Consumer Electronics (ICCE '01)*, pp. 74–75, Los Angeles, Calif, USA, June 2001.

[5] Y. Chen, Z. Zhong, T. H. Lan, S. Peng, and K. van Zon, "Regulated complexity scalable MPEG-2 video decoding for media processors," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 8, pp. 678–687, 2002.

[6] R. Braspenning, G. de Haan, and C. Hentschel, "Complexity scalable motion estimation," in *Proc. of SPIE: Visual Communications and Image Processing 2002*, vol. 4671, pp. 442–453, San Jose, Calif, USA, 2002.

[7] S. Mietens, P. H. N. de With, and C. Hentschel, "New DCT computation technique based on scalable resources," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 34, no. 3, pp. 189–201, 2003.

[8] S. Mietens, P. H. N. de With, and C. Hentschel, "Frame reordered multi-temporal motion estimation for scalable MPEG," in *Proc. 23rd International Symposium on Information Theory in the Benelux*, Louvain-la-Neuve, Belgium, May 2002.

[9] N. Cho and S. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform," *IEEE Trans. Circuits and Systems*, vol. 38, no. 3, pp. 297–305, 1991.

[10] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Transactions of the Institute of Electronics, Information and Communication Engineers*, vol. 71, no. 11, pp. 1095–1097, 1988.

[11] D. Farin, N. Mache, and P. H. N. de With, "A software-based high-quality MPEG-2 encoder employing scene change detection and adaptive quantization," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 4, pp. 887–897, 2002.

[12] T. Kummerow and P. Mohr, *Method of determining motion vectors for the transmission of digital picture information*, EPO 496 051, European Patent Application, November 1991.

[13] M. Chen, L. Chen, and T. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 5, pp. 504–509, 1994.

[14] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[15] J. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369–377, 1998.

[16] P. N. H. de With, "A simple recursive motion estimation technique for compression of HDTV signals," in *IEE 4th International Conference on Image Processing and Its Applications (IPA '92)*, pp. 417–420, Maastricht, The Netherlands, April 1992.

[17] F. Rovati, D. Pau, E. Piccinelli, L. Pezzoni, and J. M. Bard, "An innovative, high quality and search window independent motion estimation algorithm and architecture for MPEG-2 encoding," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 697–705, 2000.

[18] S. Mietens, P. H. N. de With, and C. Hentschel, "Computational complexity scalable motion estimation for mobile MPEG encoding," *IEEE Transactions on Consumer Electronics*, 2002/2003.

Stephan Mietens was born in Frankfurt (Main), Germany in 1972. He graduated in Computer Science from the Technical University of Darmstadt, Germany, in 1998 on the topic of "asynchronous VLSI design." Subsequently, he joined the University of Mannheim, where he started his research on "flexible video coding and architectures" in cooperation with Philips Research Laboratories in Eindhoven, The Netherlands. He joined the Eindhoven University of Technology in Eindhoven, The Netherlands, in 2000, where he is working towards a Ph.D. degree on "scalable video systems." Since 2003, he became a Scientific Researcher at Philips Research Labs. in the Storage and System Applications group, where he is involved in projects to develop new coding techniques.

Peter H. N. de With obtained his M.S. engineering degree from the University of Technology in Eindhoven in 1984 and his Ph.D. degree from the University of Technology Delft, The Netherlands in 1992. From 1984 to 1993, he joined the Magnetic Recording Systems Department, Philips Research Labs. in Eindhoven, and was involved in several European projects on SDTV and HDTV recording. He also contributed as a principal coding expert to the DV digital camcording standard. In 1994, he joined the TV Systems group, where he was leading advanced programmable architectures design as Senior TV Systems Architect. In 1997, he became a Full Professor at the University of Mannheim, Germany, in the Faculty of Computer Engineering. In 2000, he joined CMG Eindhoven as a principal consultant and he became a Professor in Electrical Engineering Faculty, University of Technology Eindhoven (EE Faculty). He has written numerous papers on video coding, architectures, and their realization. He is a Regular Teacher of postacademic courses at external locations. In 1995 and 2000, he coauthored papers that received the IEEE CES Transactions Paper Award. In 1996, he obtained a company Invention Award. Mr. de With is an IEEE Senior Member, Program Member of the IEEE CES (Tutorial Chair, Program Chair) and Chairman of the Benelux Information Theory Community.

Christian Hentschel received his Dr.-Ing. (Ph.D.) in 1989 and Dr.-Ing. habil. in 1996 from Braunschweig University of Technology, Germany. He worked on digital video signal processing with focus on quality improvement. In 1995, he joined Philips Research Labs. in Briarcliff Manor, USA, where he headed a research project on moiré analysis and suppression for CRT-based displays. In 1997, he moved to Philips Research Labs. in Eindhoven, The Netherlands, leading a cluster for programmable video architectures. He got the position of a Principal Scientist and coordinated a project on scalable media processing with dynamic resource control between different research laboratories. Since August 2003, he is a Full Professor at the University of Technology in Cottbus, Germany, where he heads the Department of Media Technology. He is a member of the Technical Committee of the International Conference on Consumer Electronics (IEEE) and a member of the FKTG in Germany.

EURASIP Journal on Applied Signal Processing 2004:2, 253–264

Interactive Video Coding and Transmission over Heterogeneous Wired-to-Wireless IP Networks Using an Edge Proxy

Yong Pei

Computer Science and Engineering Department, Wright State University, Dayton, OH 45435, USA
Email: ypei@cs.wright.edu

James W. Modestino

Electrical and Computer Engineering Department, University of Miami, Coral Gables, FL 33124, USA
Email: jmodestin@miami.edu

Received 26 November 2002; Revised 19 June 2003

Digital video delivered over wired-to-wireless networks is expected to suffer quality degradation from both packet loss and bit errors in the payload. In this paper, the quality degradation due to packet loss and bit errors in the payload are quantitatively evaluated and their effects are assessed. We propose the use of a concatenated forward error correction (FEC) coding scheme employing Reed-Solomon (RS) codes and rate-compatible punctured convolutional (RCPC) codes to protect the video data from packet loss and bit errors, respectively. Furthermore, the performance of a joint source-channel coding (JSCC) approach employing this concatenated FEC coding scheme for video transmission is studied. Finally, we describe an improved end-to-end architecture using an edge proxy in a mobile support station to implement differential error protection for the corresponding channel impairments expected on the two networks. Results indicate that with an appropriate JSCC approach and the use of an edge proxy, FEC-based error-control techniques together with passive error-recovery techniques can significantly improve the effective video throughput and lead to acceptable video delivery quality over time-varying heterogeneous wired-to-wireless IP networks.

Keywords and phrases: video transmission, RTP/UDP/IP, RS codes, RCPC codes, JSCC, edge proxy.

1. INTRODUCTION

With the emergence of broadband wireless networks and the increasing demand for multimedia transport over the Internet, wireless multimedia services are expected to be widely deployed in the near future. Many multimedia applications will require video transmission over links with a wireless first and/or last hop as illustrated in Figure 1. However, many existing wired and/or wireless networks cannot provide guaranteed quality of service (QoS), either because of congestion, or because temporally high bit-error rates cannot be avoided during fading periods. Channel-induced losses, including packet losses due to congestion over wired networks as well as packet losses and/or bit errors due to transmission errors on a wireless network, require customized error resilience and channel coding strategies that add redundancy to the coded video stream at the expense of reduced source coding efficiency or effective source coding rates, resulting in compromised video quality.

In this paper we quantitatively investigate the effects of packet losses on reconstructed video quality caused by bit errors anywhere in the packet in a wireless network if only error-free packets are accepted, as well as the effects of residual bit errors in the payload if errored packets are accepted instead of being discarded in the transport layer. The former corresponds to the use of the user datagram protocol (UDP) employing a checksum mechanism while the latter corresponds to the use of a transparent transport protocol, such as UDP-Lite [1], together with forward error correction (FEC) to attempt to correct transmission errors.

This work represents an extension of previous works [2, 3]. In particular, in [2] we described an approach using edge proxies which did not address the unique FEC requirements on the wired networks. This was followed by work reported in [3] where a concatenated channel coding approach was employed, but without an edge proxy, which attempted to address the distinct FEC requirements of both the wired and wireless networks.

A joint source-channel coding (JSCC) approach has been well recognized as an effective and efficient strategy to provide error-resilient image [4, 5, 6, 7, 8] and video [3, 9, 10, 11] transport over time-varying networks, such as wireless IP

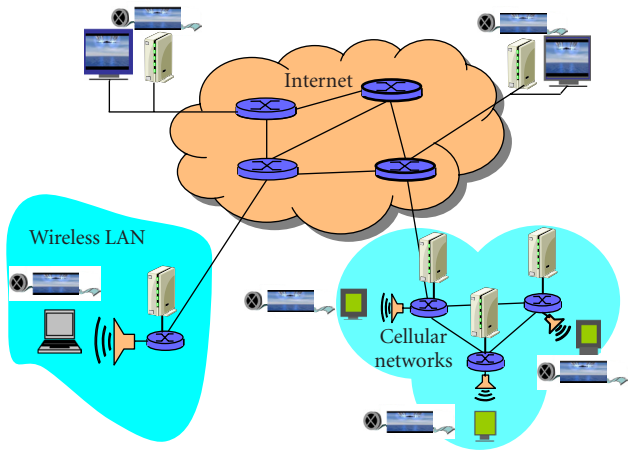


Figure 1: Illustration of heterogeneous wired-to-wireless networks.

networks. In this paper, we extend the work in [3] and provide a quantitative evaluation of a proposed JSCC approach used with a concatenated FEC coding scheme employing Reed-Solomon (RS) block codes and RCPC codes to actively protect the video data from the different channel-induced impairments associated with transmission over tandem wired and wireless networks. However, we demonstrate that this approach is not optimal since the coding overhead required on the wired link must also be carried on the wireless link which can have a serious negative effect on the ability of the bandwidth-limited wireless link to support high-quality video transport.

Finally, we will present a framework for an end-to-end solution for packet video over heterogeneous wired-to-wireless networks using an edge proxy. Specifically, the edge proxy serves as an agent to enable and implement selective packet relay, error-correction transcoding, JSCC, and interoperation between different transport protocols for the wired and wireless networks. Through the use of the edge proxy located at the boundary of the wired and wireless networks, we demonstrate the ability to avoid the serious compromise in efficiency on the wireless link associated with the concatenated approach. More specifically, we employ RS codes only on the wired network to protect against packet losses while the RCPC codes are employed only on the wireless network to protect against bit errors. The edge proxy provides the appropriate FEC transcoding resulting in improved bandwidth efficiencies on the wireless network. We believe that the value of the proposed approach, employing an edge proxy with appropriate functionalities, lies in the fact that little or no change needs to be provided on the existing wired network while at the same time it addresses the distinctly different transport requirements for the wireless network. Furthermore, it uses fairly standard FEC approaches in order to support reliable multimedia services over the Internet with a wireless first and/or last hop.

The remainder of this paper is organized as follows. In Section 2, we provide some technical preliminaries describing an application level framing (ALF) approach employing RTP-H.263+ packetization. In Section 3, we briefly describe the background for packet video over wireless networks and provide a quantitative study of packet video performance over wireless networks based on the two different transport-layer strategies as discussed above. We also describe the RCPC codes, the channel-loss model, and the assumed physical channel model for the wireless networks under study. In Section 4, we introduce a concatenated FEC coding scheme for packet video transport over heterogeneous wired-to-wireless networks, and briefly describe the interlaced RS codes and packetization scheme employed. In Section 5, we present a framework for an end-to-end solution for packet video over heterogeneous wired-to-wireless network using edge proxies and provide a comparison of the performance achievable compared to the concatenated approach. Finally, Section 6 provides a summary and conclusions.

2. PRELIMINARIES

2.1. *Application-layer framing*

To provide effective multimedia services over networks lacking guaranteed QoS, such as IP-based wired as well as wireless networks, it is necessary to build network-aware applications which incorporate the varying network conditions into the application layer instead of using the conventional layered architecture to design network-based applications. A possible solution is through ALF as proposed in [12]. The principal concept of ALF is that most of the functionalities necessary for network communications will be implemented as part of the application. As a result, the underlying network infrastructure provides only minimal needed functionalities. The application is then responsible for assembling data packets, FEC coding and error recovery, as well as flow control. The protocol of choice for IP-based packet video applications is the real-time transport protocol (RTP) [13], which is an implementation of ALF by the internet engineering task force (IETF). Likewise, UDP-Lite [1] is a specific instance of ALF in the sense that the degree of transparency at the transport layer can be tailored to the application by allowing the checksum coverage to be variable, including only the header or portions of the packet payload as well. In this paper, we will consider the use of ALF-based RTP-H.263+ for video transmission over wired and wireless IP networks with a simplified transparent transport layer that does not require all the functionalities of UDP-Lite.

2.2. *RTP-H.263+*

In order to transmit H.263+ video over IP networks, the H.263+ bitstream must first be packetized. A payload format for H.263+ video has been defined for use with RTP (RFC 2429) [14]. This payload format for H.263+ can also be used with the original version of H.263. In our experiments, the group of block (GOB) mode was selected for the H.263+ coder and packetization was always performed at GOB boundaries, that is, each RTP packet contains one

or more complete GOBs. Since every packet begins with a picture or GOB start code, the leading 16 zeros are omitted in accordance with RFC 2429 [14]. The packetization overhead then consists only of the RTP/UDP/IP headers, which are typically 40 bytes per packet. This overhead can be significant at low bit rates for wireless network-based applications. It is important to improve the packetization efficiency in such cases [15]. To minimize the packetization header overhead, each RTP packet should be as large as possible. On the other hand, in the presence of channel impairments, the packet size should be kept small to minimize the effects of lost packets on reconstructed video quality.

3. PACKET VIDEO OVER WIRELESS NETWORKS

Knowledge of the radio propagation characteristics is usually a prerequisite for effective design and operation of a communication system operating in a wireless environment. The fading characteristics of different radio channels and their associated effect on communication performance have been studied extensively in the past [16]. Despite the fact that Rayleigh fading is the most popular model, Rician fading is observed in mobile radio channels as well as in indoor cordless telecommunication (CT) systems [16]. In a cellular system, Rayleigh fading is often a feature of large cells, while for cells of smaller diameter, the envelope fluctuations of a received signal are observed to be closer to Rician fading. A slow and flat Rician fading model is assumed here,[1] where the duration of a symbol waveform is sufficiently short so that the fading variations cause negligible loss of coherence within each received symbol. At the same time, the symbol waveform is assumed to be sufficiently narrowband (sufficiently long in duration) so that frequency selectivity is negligible in the fading of its spectral components. As a result, the receiver can be designed and analyzed on the basis of optimal symbol-by-symbol processing of the received waveform, for example, by a sampled matched filter or other appropriate substitute in the same manner used in the nonfading case.

3.1. Channel-induced loss models

In this work, we restrict our attention to a random loss model, that is, the wireless channel is characterized by uncorrelated bit errors. This is a reasonable model for a fairly benign wireless channel under the assumption of sufficient interleaving to randomize the burst errors produced in the decoder.

By means of FEC, some of these bit errors can be corrected. Depending on the FEC code parameters and the channel conditions, there will be residual bit errors. Generally, over existing wired IP networks, UDP is configured to discard any packet with even a single error detected in the entire packet including the header, although UDP itself need not implement this error-detecting functionality. In the wireless video telephony system described by Cherriman et al. [17], such packets are also discarded without further processing. In this paper, we will define two channel-induced loss models. For the first model, we assume the same loss model as used in wired IP networks; that is, a packet is accepted only if there is no error in the entire packet including the header as well as the payload, otherwise, it is considered lost. This model corresponds to a transport scheme allowing only error-free packets (denoted as scheme 1 in this paper). So, for an interference-limited wireless channel, like the CDMA radio interface, the packet losses are primarily the results of frequent bit errors instead of congestion as in a wired network. The channel-induced impairment to the video quality is in the form of these packet losses. If a packet is considered lost, the RTP sequence number enables the decoder to identify the lost packets so that locations of the missing GOBs are known. The missing blocks can then be concealed by motion-compensated interpolation using the motion vector of the macroblock (MB) immediately above the lost MB in the same frame, or else the motion vector is assumed to be zero if this MB is missing. However, if too many packets are lost, concealment itself is no longer effective in improving the reconstructed video quality.

For the second model, we assume that the transport layer is transparent to the application layer; that is, a packet with errors only in the payload is not simply discarded in the transport layer. Such a transparent transport layer can be achieved by using, for example, UDP-Lite as proposed in [1]. However, UDP-Lite provides other functionalities not necessary for the work here and is not widely deployed. As a result, we employ a simplified transparent transport protocol which limits the use of the checksum only on the RTP/UDP/IP header and discards a packet only if there is an error detected in the header. In this case the application layer should be able to access the received data although such data may have one or more bit errors. This model corresponds to a transport scheme allowing bit errors in the payload (denoted as scheme 2 in this paper). The channel-induced impairment to the video quality is then in the form of residual bit errors in the video stream. It is the responsibility of the application layer to deal with these possible bit errors. Specifically, here we make use of the H.263+ coding scheme where, based on syntax violations, certain error patterns may be detected by the video decoder and the use of the corresponding errored data can be avoided by employing passive error-recovery (PER) techniques.

Our intention is to quantitatively compare these two channel-induced loss models, identify the different video data protection requirements for wired and wireless networks, and describe the corresponding appropriate transport schemes for packet video delivery over such networks.

3.2. Physical channel model

The bitstreams are modulated before being transmitted over a wireless link. During transmission, the modulated bitstreams typically undergo degradation due to additive white

[1] The slow and flat Rician channel model is completely described in terms of the single parameter ζ^2 representing the ratio of specular-to-diffuse energy.

Gaussian noise (AWGN) and/or fading. At the receiver side, the received waveforms are demodulated, channel decoded, and then source decoded to form the reconstructed video sequence. The reconstructed sequence may differ from the original sequence due to both source coding errors and possible channel-error effects.

In this paper, the symbol transmission rate for the wireless links is set to be $r_S = 64$ Ksps, such that the overall bit rate employing QPSK modulation is constrained as $R_{\text{tot}} = 128$ Kbps. This in turn sets the upper limit for the bit rate over the wired networks to be $R_{\text{tot}} = 128$ Kbps as well. Since the total bit rate is limited by the wireless links, the use of RS and/or RCPC codes will result in a decrease of source coded bit rate proportional to the overall channel coding rates.

The transmission channel is modelled as a flat-flat Rician channel with ratio of specular-to-diffuse energy $\zeta^2 = 7$ dB.

3.3. RCPC channel codes

The class of FEC codes employed for the wireless IP network in this work is the set of binary RCPC codes described in [18]. With P representing the puncturing period of the code, the rates of the codes that may be generated by puncturing a rate $R_c = 1/n$ mother code are $R_c = P/(P + j)$, $j = 1, 2, \ldots, (n-1)P$. Thus, it is easy to obtain a family of codes with unequal error correcting capabilities. In this work, a set of RCPC codes are obtained by making use of an $R_c = 1/4$ mother code with memory $M = 10$ and a corresponding puncturing period $P = 8$. Then the available RCPC codes are of rates, $R_c = 8/9, 8/10, \ldots, 8/32$.

3.4. Passive error recovery

If a packet is considered lost, the RTP sequence number enables the decoder to identify the lost packets so that locations of the missing data are known. The affected blocks can then be concealed by PER techniques. In this work, we make use of the error-detecting and recovery scheme described in Test Model 8 [19]. The major objective of this PER scheme is to detect the severe error patterns and prevent the use of such errors which may substantially degrade the video quality. The remaining undetected error patterns in the payload which are not detected by the H.263+ decoder will result in the use of incorrectly decoded image data which can cause quality degradation of the reconstructed video.

3.5. Selected simulation results

We present some selected results for a representative quarter common intermediate format (QCIF) video conferencing sequence, Susie at 7.5 frames per second (fps). These results were obtained using a single-layer H.263+ coder in conjunction with RCPC channel codes [18] together with quadrature phase shift keyed (QPSK) modulation. To decrease the sensitivity of our results to the location of bit errors, a sequence of $N_f = 30$ input frames is encoded, channel errors are simulated and the resulting distortion is averaged. Furthermore, each simulation was run N_t times. By taking empirical averages with N_t sufficiently large (i.e., $N_t = 1000$), statistical confidence in the resulting distortion can be achieved.

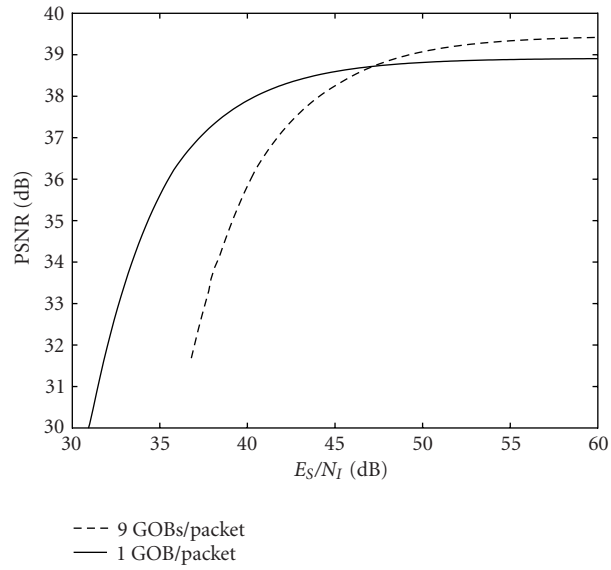


Figure 2: Performance of RTP-H.263+ packet video with 1 or 9 GOBs/packet over a wireless channel without channel coding and employing loss model 1; Rician channel with $\zeta^2 = 7$ dB.

Figure 2 demonstrates results for a system without channel coding under the assumption of the first loss model. Here, we plot the reconstructed peak signal-to-noise ratio (PSNR) versus the channel SNR, E_S/N_I.[2] In Figure 2, we provide results for two packetization choices which packetize either 1 or 9 GOBs (i.e., 1 frame for QCIF) into a single packet. It should be obvious that in the absence of channel impairments, the more GOBs contained in one packet, the better the quality should be as a result of the reduced overheads. This is clearly demonstrated in Figure 2 where for large E_S/N_I, the larger number of GOBs/packet results in improved PSNR performance. However, as the channel conditions degrade (i.e., the value of E_S/N_I decreases), a packetization scheme with fewer GOBs/packet can be expected to be more robust in the presence of the increasing channel impairments. This is because of the dependence of packet-loss rate upon the corresponding packet size. Although the bit-error rate remains the same, a larger packet size results in larger packet-loss rate. This is also demonstrated in Figure 2. It should also be noticed that under the first loss model, the video quality is extremely sensitive to packet losses due to the channel variation in E_S/N_I.

Next, we demonstrate the performance of the system with a transparent transport layer; that is, channel-loss model 2. We provide corresponding results in Figure 3 for both loss models for two packetization choices which again packetize 1 or 9 GOBs (i.e., 1 frame for QCIF) into a single packet. If a single GOB is packetized into a packet, the quality of the second transport scheme degrades somewhat

[2] The quality E_S/N_I represents the ratio of energy per symbol to the spectral density of the channel noise or interference level.

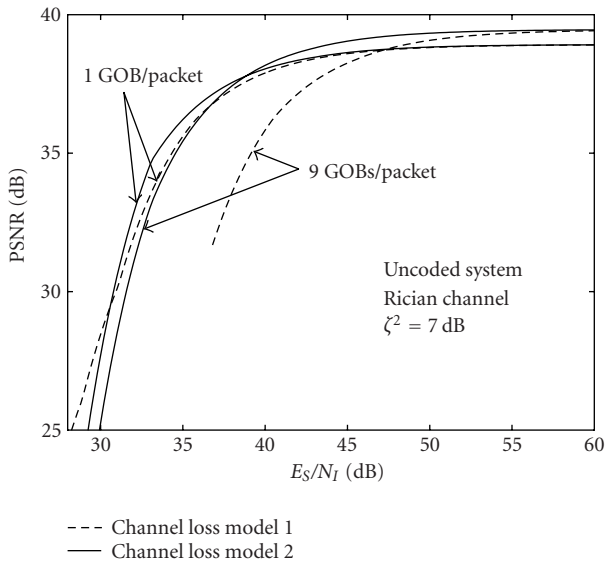


Figure 3: Performance of RTP-H.263+ packet video with 1 or 9 GOBs/packet over a wireless channel without channel coding for the two loss models.

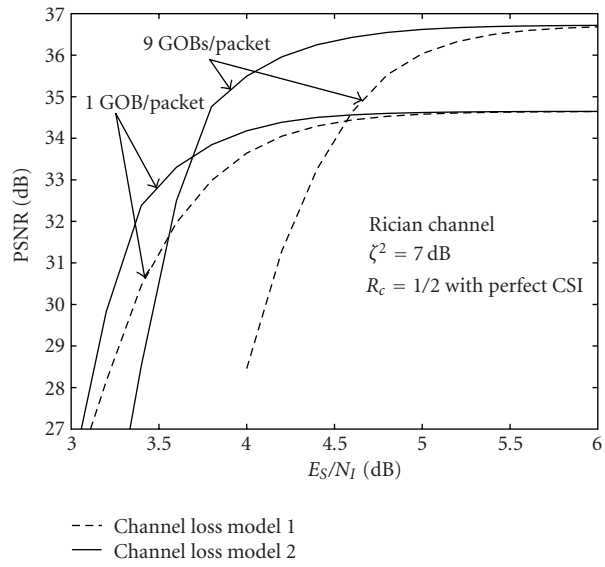


Figure 4: Performance of RTP-H.263+ packet video with 1 or 9 GOBs/packet over a wireless channel with a fixed $R_c = 1/2$, $M = 10$ convolutional code for the two loss models.

more gracefully compared to the first scheme as the channel E_S/N_I decreases. The relative disadvantage of the first scheme in this case is the result of discarding packets with even a single bit error in the payload. Instead, the second scheme makes use of the received data by selectively decoding those data without severely degrading the video quality. Since the packet size in this case is relatively small, as the bit error rate increases as a result of decreasing E_S/N_I, there is some advantage of the first scheme in the region $E_S/N_I < 31$ dB because it avoids the use of error-prone packets. For scheme 2, on the other hand, the remaining undetected errors in the payload begin to overwhelm the PER capabilities of the decoder as E_S/N_I decreases and substantially degrade the reconstructed video quality. This is also demonstrated in Figure 3. However, it should be noticed that in this region the video quality is already sufficiently degraded that the relative advantage of scheme 1 in this region does not make a significant difference for video users. Furthermore, as illustrated in Figure 3, if 9 GOBs are packetized into a packet, the quality of the second transport scheme substantially outperforms the first scheme as the channel E_S/N_I becomes smaller. As the packet size increases, the disadvantage of the first scheme is even more significant as a result of discarding packets with even single bit error in the payload. Based on these observations, it would appear that it is necessary to provide a transparent transport scheme for packet video over wireless networks. More specifically, packet video over wired and wireless IP networks may have to employ different transport-layer protocols.

FEC can be used to protect the video data against channel errors to improve the video delivery performance in the range of lower E_S/N_I, although, as we demonstrate, the choice of channel coding rate must be carefully made. For example, the corresponding results for the previous two packetization choices are illustrated in Figure 4 for the two loss models where we somewhat arbitrarily employ an $R_c = 1/2$, $M = 10$ convolutional code to protect the packetized video data. In this case, the additional channel coding overheads force a decrease in the available source coding bit rate,[3] and this results in a corresponding decrease in the video quality in the absence of channel impairments. This can be seen if we compare the results in Figure 4 to the corresponding values in Figure 3 for large E_S/N_I. However, it should be noted that the coded cases can maintain the video quality at acceptable levels for considerably smaller values of E_S/N_I compared to the uncoded system. This is a good indication of the necessity of employing FEC coding in wireless networks.

It should also be observed in Figure 4, compared to the uncoded case illustrated in Figure 3, that the second loss model consistently and substantially outperforms the first loss model. For example, there is over 6 dB performance gain of the second model over the first model at $E_S/N_I = 4$ dB for the case of 9 GOBs/packet. This suggests the advisability of using FEC coding to constrain the bit-error rate in wireless networks together with the use of a transparent transport-layer scheme to provide acceptable packet video services. This provides further illustration that packet video transport over wireless IP networks may require a different transport-layer protocol from conventional wired networks in order to obtain more desirable error-resilient quality.

[3] Recall that we are holding the total transmitted bit budget at $R_{\text{tot}} = 128$ Kbps.

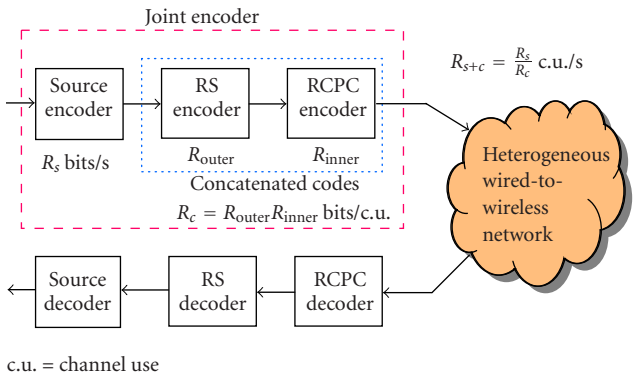


Figure 5: Illustration of concatenated coding scheme.

4. PACKET VIDEO OVER WIRED-TO-WIRELESS IP NETWORKS

Many evolving multimedia applications will require video transmission over a wired-to-wireless link such as in wireless IP applications where a mobile terminal communicates with an IP server through a wired IP network in tandem with a wireless network as illustrated in Figure 1. We intend to address an end-to-end solution for video transmission over a heterogeneous network such as the UMTS third-generation (3G) wireless system, which provides the flexibility at the physical layer to introduce service-specific channel coding as well as the necessary bit rate required for high-quality video up to 384 Kbps.

Video quality should degrade gracefully in the presence of either packet losses due to congestion on the wired network, or bit errors due to fading conditions on the wireless channel. Due to the difference in channel conditions and loss patterns between the wired and wireless networks, to be efficient and effective the error-control schemes should be tailored to the specific characteristics of the loss patterns associated with each network. Furthermore, the corresponding error-control schemes for each network should not be designed and implemented separately, but jointly in order to optimize the quality of the delivered video.

Here, we present a possible end-to-end solution which employs an adaptive concatenated FEC coding scheme to provide error-resilient video service over tandem wired-to-wireless IP networks as illustrated in Figure 5. An H.263+ source coder encodes the input video which is applied to a concatenated channel encoder employing an RS block outer encoder and an RCPC inner encoder. The RS outer code operates in an erasure-decoding mode and provides protection against packet loss due to congestion in the wired IP network while the RCPC inner code provides protection against bit errors due to fading and interference on the wireless network. The RS coding rates can be selected adaptively according to the prevailing network conditions, specifically, packet-loss rate for the wired IP network. This channel rate matching is achieved by employing a set of RS codes with different erasure-correcting capabilities. The RCPC coding rates can also be selected adaptively to provide different levels of bit-error-correcting capability according to the prevailing wireless network conditions, specifically, E_S/N_I for the wireless channels.[4] This end-to-end approach avoids the system complexities associated with transcoding in edge proxies located at the boundaries between the wired and wireless networks as treated in [2], for example. However, we will see that this reduction in complexity is at the expense of a considerable performance penalty.

4.1. Packet-level FEC scheme for wired IP networks

Packet loss is inevitable even in wired IP networks, and can substantially degrade reconstructed video quality which is annoying for users. Thus, it is desirable that a video stream be robust to packet loss. Regarding the tight delay constraints for real-time video applications, FEC should be applied to achieve error recovery when packet losses occur. For a wired IP network, packet loss is caused primarily by congestion, and channel coding is typically used at the packet-level [20, 21] to recover from such losses. Specifically, a video stream is first chopped into segments each of which is packetized into k packets, and then for each segment, a block code is applied to the k packets to generate an n-packet block, where $n > k$. To perfectly recover a segment, a user only needs to receive any k packets in the n-packet block. To avoid additional congestion problems due to channel-coding overheads, a JSCC approach to optimize the rate allocation between source and channel coding is necessary. One such approach employing interlaced RS coding with packet-loss-recovery capability has been described in [22].

In this paper, we will apply a form of concatenated FEC coding employing interlaced RS codes as illustrated in Figure 6, where FEC codes are applied across IP packets. Specifically, each packet is partitioned into successive m-bit symbols to form an encoding array, and individual symbols are aligned vertically to form RS codewords of block length n over GF(2^m). As illustrated in Figure 6, each IP packet consists of w successive rows of m-bit symbols, then, the decoded packet-loss probabilities can be readily determined assuming erasure-only decoding.

4.2. Packetization for the interlaced RS coded video data

To quantitatively compare the performance between a coded system and an uncoded system, we have to maintain the same packet-generation rate. Specifically, for the QCIF video studied in this paper, in the uncoded system, each GOB is packetized into a single packet, resulting in 9 packets per video frame. For the coded system, network packets are obtained by concatenating successive rows of the encoding array illustrated in Figure 6. We maintain identical packet rate in the coded system as in the uncoded system. Specifically, with the use of RS$(63, k)$ codes, this results in packing 7 (i.e., $w = 7$ in Figure 6) coded symbols from the same RS codeword into the same packet together with other RS coded symbols from

[4] The RCPC rates should also depend on the Rician channel parameter ζ^2 which for purposes of this work we will assume is fixed and known.

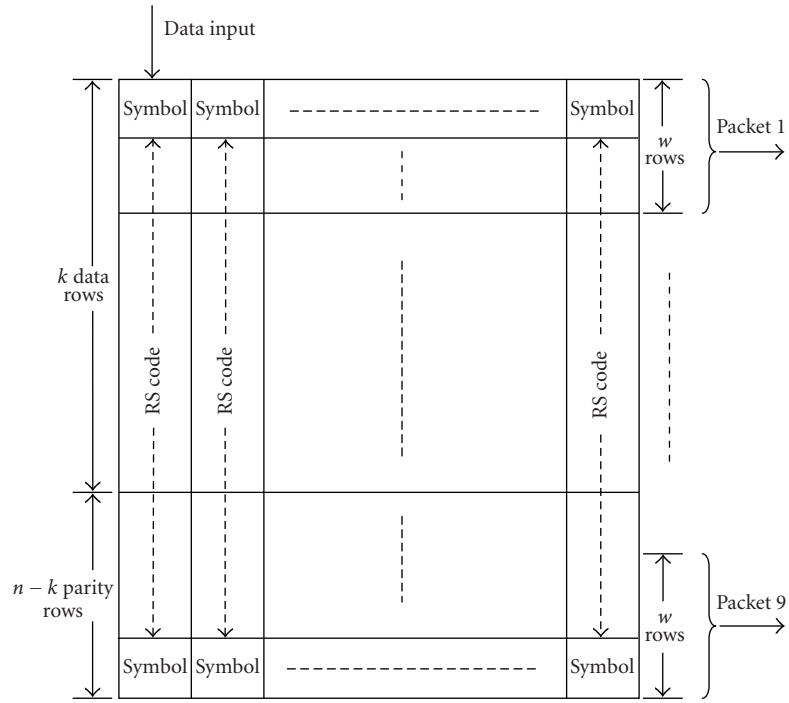


Figure 6: Illustration of interlaced RS codes.

the same video frame. As a result, both systems will generate 9 packets per frame.

4.3. Packet-loss correction using RS codes

Consider an RS(n, k) code over GF(2^m) applied in an interlaced fashion across the IP packets as described above and illustrated in Figure 6. Here, k symbols of m bits each are encoded into n m-bit symbols with d the minimum distance of the RS code given by

$$d = n - k + 1. \tag{1}$$

For the proposed concatenated FEC scheme, it is possible that there are residual bit errors that cannot be corrected through the use of the inner RCPC codes. These residual bit errors may degrade the erasure-correction capability of the RS codes employing erasure decoding which attempts to correct the packet-loss-induced symbol erasures over the wired IP network. However, the probability of symbol errors for the RS coded symbols resulting from such residual bit errors will be very small compared to the symbol-erasure rate with appropriate choices of inner RCPC codes which maintain the residual bit-error rate low. For example, considering an RS$(63, k)$ code with a symbol size of 6 bits, a residual bit-error rate of 10^{-5} will result in a symbol-error rate of 6×10^{-5} which will have a negligible effect on the erasure correcting performance of the RS codes in a system where packet-loss-induced erasures are dominant. So, in this paper we assume the use of erasure-only decoding of RS codes with full erasure-correcting capability.

For an RS code with erasure decoding, $e \leq d - 1$ erasures can be corrected. Consider that w m-bit symbols from an RS codeword are packed into the same packet. A packet loss under this packetization scheme will result in w erasures for the corresponding RS coded symbols. Assume the symbol erasures are independent. For the coded system, the resulting packet-loss rate for the above specified packetization scheme then becomes

$$\mathrm{P_L} = \sum_{i=W}^{9} \binom{9}{i} \lambda^i (1-\lambda)^{9-i}, \tag{2}$$

where λ is the corresponding uncoded packet-loss rate, and W is the maximum number of allowable packet losses that can be recovered through the use of RS codes, and is given by

$$W = \lceil e/w \rceil. \tag{3}$$

It should be noted that a lost packet in the uncoded system as described above will result in a loss of 1 GOB. However, for the coded system, if there is a packet loss that cannot be recovered through the erasure-correcting capability of the corresponding RS codes, the whole frame, that is 9 GOBs, will be affected due to the interlaced RS coding scheme. In such a situation, PER, as will be described in Section 4.4, will be applied to conceal the errors.

4.4. Channel-induced loss models

In the previous section, we have shown the advantage of a transparent transport layer for video transmission over noisy wireless channels. In what follows, we will again assume that

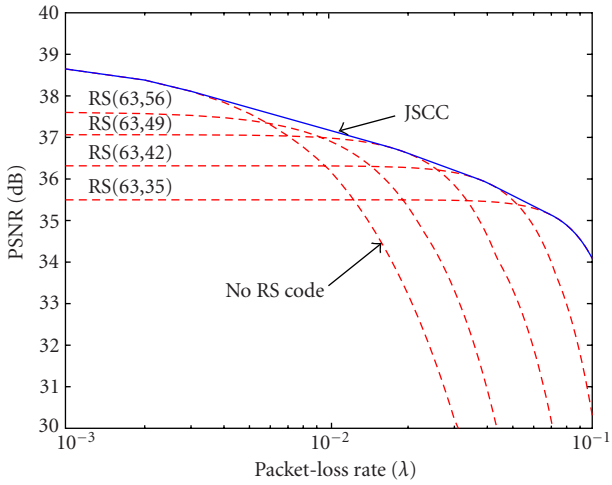


Figure 7: Performance of RTP-H.263+ packet video over wired IP networks using RS coding alone.

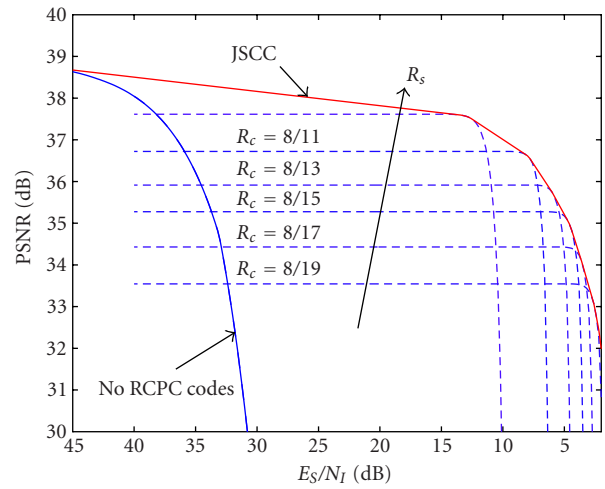


Figure 8: Performance of H.263+ coded video delivery over a wireless Rician fading channel with $\zeta^2 = 7$ dB using JSCC approach with RCPC coding only and employing perfect CSI. Performance results for a set of fixed channel coding rate schemes are also shown.

the transport layer is transparent to the application layer, that is, a packet with errors in the payload is not simply discarded in the transport layer. Instead, the application layer should be able to access the received data although such data may have one or more bit errors. It is the responsibility of the application layer to deal with the possible residual bit errors as described previously in Section 3.1.

4.5. JSCC approach

As has been demonstrated in the previous section, in order to protect against the channel impairments, some form of FEC coding must be employed. Since an arbitrarily chosen FEC design can lead to a prohibitive amount of overhead for highly time-varying error conditions over wireless channels, a JSCC approach for image or video transmission is necessary. The objective of JSCC is to jointly select the source and channel coding rates to optimize the overall performance due to both source coding loss and channel-error effects subject to a constraint on the overall transmission bit rate budget.

In [9, 10], it was shown that much of the computational complexity involved in solving this optimal rate allocation problem may be avoided through the use of universal distortion rate characteristics. Given a family of universal distortion rate characteristics for a specified source coder, together with appropriate bounds on bit-error probability P_b for a particular modulation/coding scheme as a function of channel parameters, the corresponding optimal distortion rate characteristics for a video sequence can be determined through the following procedure: for a specified channel SNR, E_S/N_I, we can find the associated P_b through the corresponding bit-error probability bounds for a selected modulation/coding scheme as discussed earlier. Then, for each choice of source coding rate R_s of interest, use the resulting P_b to find the corresponding overall PSNR from the universal distortion rate characteristics. This procedure is described in more detail in [9, 10].

4.6. Selected simulation results

We first consider the case where no channel error is introduced over the wireless links; that is, only the packet loss over the wired network will degrade the video quality. Figure 7 demonstrates the performance using a family of RS$(63, k)$ codes[5] with JSCC for RTP-H.263+ packet video over wired IP networks experiencing random packet loss. Here we illustrate PSNR results as a function of packet-loss rate λ for different values of source coding rate with the RS codes chosen to achieve the overall bit rate budget $R_{\text{tot}} = 128$ Kbps. In particular, the smaller values of R_s allow the use of more powerful low-rate RS codes resulting in improved performance for larger packet-loss rate. On the other hand, for small packet-loss rate performance, improvements can be obtained using larger values of R_s together with less powerful high-rate RS codes. The optimum JSCC procedure selects the convex hull of all such operating points as illustrated schematically in Figure 7. Clearly, compared to the system without using RS coding where video quality is substantially degraded with increasing packet-loss rate, the JSCC approach with RS coding provides an effective means to maintain the video quality as network-induced packet-loss rate increases.

Consider another case where now bit errors over the wireless links instead of packet loss over the wired network are dominant, and a JSCC approach using RCPC codes is employed. The results are illustrated in Figure 8 where we now plot PSNR versus E_S/N_I.[6] Again, as can be observed, the JSCC approach with RCPC coding alone clearly demonstrates significant performance improvements over either the uncoded case or the case where the channel coding rate is fixed at

[5] RS$(63, k)$ codes are used throughout the remainder of this paper.

[6] Observe the decreasing values of E_S/N_I used in plotting Figure 8.

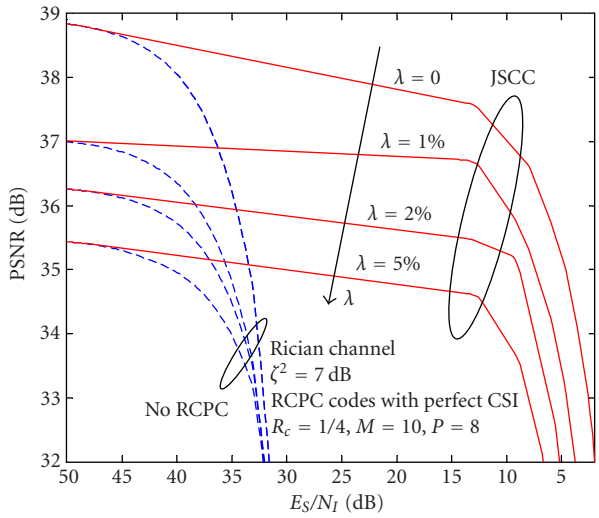


FIGURE 9: Performance of H.263+ coded video delivery over heterogeneous wired-to-wireless IP networks using JSCC employing concatenated RS and RCPC coding.

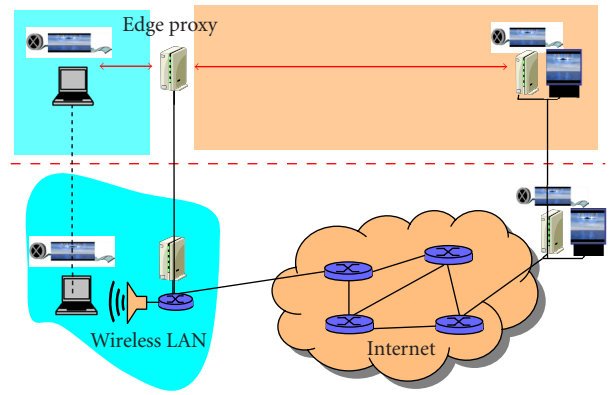


FIGURE 10: An end-to-end approach using an edge proxy.

an arbitrarily chosen value.[7] The use of JSCC can provide a more graceful pattern of quality degradation by keeping the video quality at an acceptable level for a much wider range of E_S/N_I. This is achieved by jointly selecting the channel and source coding rates based on the prevailing channel conditions, here represented by E_S/N_I.

In more general cases, packet loss due to congestion in the wired network and bit errors due to fading effects on the wireless networks coexist. We propose to jointly select the source coding rate, the RS coding rate, and the RCPC coding rate such that optimal end-to-end performance can be achieved with this concatenated coding scheme. Here, we demonstrate PSNR results for reconstructed video as a function of the wireless channel E_S/N_I for a set of packet-loss rates over the wired IP network with the RS codes and RCPC codes chosen to achieve the overall bit rate budget $R_{\text{tot}} = R_s/(R_c^{\text{RCPC}} \cdot R_c^{\text{RS}}) = 128$ Kbps [3]. In Figure 9, for a given packet-loss rate λ in the wired network, the optimal performance obtainable is demonstrated under the constraint of a fixed wireless transmission rate. It is clear that the RS coding rate has to be adaptively selected with the variation in the corresponding packet-loss rate. Meanwhile, the RCPC coding has to adapt to the change in the wireless link conditions, E_S/N_I in this case. Clearly, as shown by the dashed lines in Figure 9, for the system employing only adaptive RS codes selected according to the packet-loss rate on the wired network but no RCPC codes on the wireless network, video quality is substantially degraded with increasing bit errors as E_S/N_I decreases. In contrast, the JSCC approach with concatenated RS and RCPC coding provides an effective means to maintain the video quality as network-induced packet-loss and/or bit-error rate increase.

5. PACKET VIDEO OVER WIRED-TO-WIRELESS IP NETWORK USING AN EDGE PROXY

In the previous section, we investigated a JSCC approach used with a concatenated FEC coding scheme employing interlaced RS block codes and RCPC codes to actively protect the video data from different channel-induced impairments over tandem wired and wireless networks. However, this approach is not optimal since, as noted previously, the coding overhead required on the wired link must also be carried on the wireless link.

As an alternative to the concatenated approach, we present an end-to-end solution with the use of an edge proxy operating at the boundary of the two networks as demonstrated in Figure 10. This end-to-end solution employs the edge proxy to enable the use of distinctly different error-control schemes on the wired and wireless networks. Specifically, we employ the interlaced RS codes alone on the wired network and the RCPC codes alone on the wireless network to provide error-resilient video service over tandem wired-to-wireless IP networks. As a result, under the constraint of a total bitrate budget R_{tot}, the effective video data throughput is given as $R_s = \min\{R_{\text{tot}} \cdot R_c^{\text{RS}}, R_{\text{tot}} \cdot R_c^{\text{RCPC}}\}$, where R_c^{RS} and R_c^{RCPC} are the channel coding rates for the RS and RCPC codes, respectively. In contrast, without the use of an edge proxy, these two codes have to work as a concatenated FEC scheme as described in the preceding section in order to provide sufficient protection against both congestion-caused packet loss in the wired network and fading-caused bit errors in the wireless network. The corresponding effective video data throughput in this case is then $R_s = R_{\text{tot}} \cdot R_c^{\text{RS}} \cdot R_c^{\text{RCPC}}$ and, because of the need to carry both overheads on both networks, this causes a serious reduction in achievable video quality. It is clear then that the reconstructed video quality can be improved through the use of an edge proxy. We will quantitatively investigate the resulting improvement for interactive video coding and transmission in what follows.

[7] For example, the arbitrary choice of $R_c = 1/2$ illustrated in Figure 4 would fall between the curves labelled $R_c = 8/15$ and $R_c = 8/17$ in Figure 8.

5.1. Edge proxy

To accommodate the differential error-control schemes as well as differential transport protocols for packet video over wired and wireless networks, appropriate middleware has to be employed to operate between the wired and wireless network to support the application layer solutions for video applications. Thus, we define an edge proxy here to accomplish these functionalities. The edge proxy should be implemented as part of a mobile support station. Furthermore, it should be application-specific; in our case it is video-oriented.

The use of edge proxies at the boundaries of dissimilar networks for a variety of functions have been used extensively in the networking community [23]. The uniqueness of the approach proposed here using edge proxies at the boundary between wired and wireless networks for video transport applications lies in its specific functionalities as defined above. Specifically, it serves as an agent to enable and implement

(1) selective packet relay,
(2) error-control transcoding,
(3) JSCC control,
(4) interoperation between different possible transport protocols for the wired and wireless network.

For the interactive applications we consider here, there exists two-way traffic including wired-to-wireless as well as wireless-to-wired. We assume that RS codes are employed to combat packet loss due to congestion in a wired network, and RCPC codes are used on the wireless network to combat bit errors. It is necessary for the edge proxy to do error-control transcoding if such a scheme is used.

Furthermore, the edge proxy should support the JSCC control scheme to adaptively adjust the source and channel coding rates. To avoid computation and time-expensive video transcoding in the edge proxy, an end-to-end adaptive coding control strategy is suggested here. The channel conditions including those for both the wired and wireless networks are collected in the edge proxy, and based on the prevailing channel conditions, video coding rates are adjusted accordingly using JSCC. For the wired network, the major channel condition parameter is the packet-loss rate, while for the wireless network, channel SNR as well as the fading parameters are used.

The edge proxy is also responsible for the interoperation between different possible transport protocols for the wired and wireless network. For a wireless network, the error-control scheme is implemented in the application layer, and erroneous packets should be delivered to the end user. However, for conventional wired networks, such as existing IP networks, no error is allowed. In this case, to achieve interoperation, the edge proxy has to repacketize the packet according to the appropriate transport protocol before relaying the packet in either direction.

5.2. Selected simulation results

Now we consider the system with the use of an edge proxy between the wired and wireless IP networks, such that error-control transcoding can be done between the two heterogeneous networks each supporting different error-control approaches as described previously. With the use of an edge proxy, the corresponding optimal performance obtainable is demonstrated in Figure 11 under the constraint of the same fixed wireless transmission rate of 128 Kbps.

For comparison, we also present in Figure 12 the results for the systems with or without the use of an edge proxy under the same transmission rate limit, which have been shown previously in Figures 11 and 9, respectively. It clearly

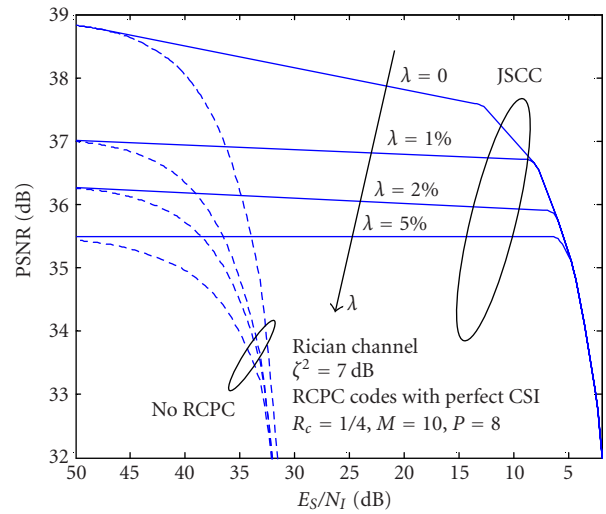


Figure 11: Performance of H.263+ coded video delivery over heterogeneous wired-to-wireless IP networks using JSCC with an edge proxy.

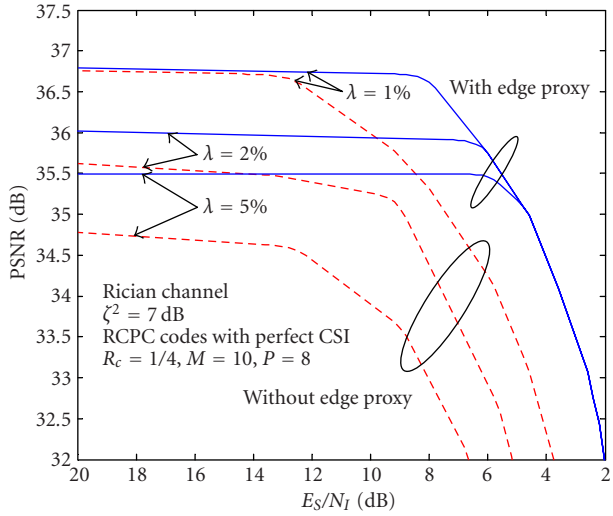


Figure 12: Relative performance improvement with and without the use of an edge proxy.

demonstrates the substantial improvement using an edge proxy. For example, in the case that packet-loss rate over the wired IP network is $\lambda = 5\%$, there is a gain of over 6 dB in wireless channel E_S/N_I for a specified video quality of PSNR = 34 dB. This improvement is primarily due to the increase of effective video data throughput due to the error-control transcoding in the edge proxy. As a result, to meet the same error protection requirement for both wired and wireless network conditions, a larger effective video data throughput can be achieved through the use of an edge proxy compared to the case without an edge proxy.

6. SUMMARY AND CONCLUSIONS

We quantitatively demonstrate the requirements for different transport-layer schemes for packet video over wireless networks from the requirements for conventional wired networks. Then we described the possible end-to-end solutions with and without an edge proxy operating between the wired and wireless network for packetized H.263+ video over heterogeneous wired-to-wireless IP networks. A JSCC approach employing RS block codes and RCPC codes is studied for the two proposed architectures. The results quantitatively demonstrate the requirement for a joint design approach to address the special needs of error recovery for packet video over the wireless and wired network for acceptable end-to-end quality while exhibiting a graceful pattern of quality degradation in the face of dynamically changing network conditions. Furthermore, the results clearly demonstrate the advantage of using an edge proxy with clearly defined functionalities in heterogeneous wired-to-wireless IP networks for improved video quality.

REFERENCES

[1] L.-Å. Larzon, M. Degermark, and S. Pink, "UDP Lite for real time multimedia applications," in *Proc. IEEE International Conference of Communications*, Vancouver, BC, Canada, June 1999.

[2] Y. Pei and J. W. Modestino, "Robust packet video transmission over heterogeneous wired-to-wireless IP networks using ALF together with edge proxies," in *Proc. European Wireless Conference*, Florence, Italy, February 2002.

[3] Y. Pei and J. W. Modestino, "Use of concatenated FEC coding for real time packet video over heterogeneous wired-to-wireless IP networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 840–843, Bangkok, Thailand, May 2003.

[4] M. J. Ruf and J. W. Modestino, "Operational rate-distortion performance for joint source and channel coding of images," *IEEE Trans. Image Processing*, vol. 8, no. 3, pp. 305–320, 1999.

[5] P. C. Cosman, J. K. Rogers, P. G. Sherwood, and K. Zeger, "Combined forward error control and packetized zerotree wavelet encoding for transmission of images over varying channels," *IEEE Trans. Image Processing*, vol. 9, no. 6, pp. 982–993, 2000.

[6] P. G. Sherwood, X. Tian, and K. Zeger, "Efficient image and channel coding for wireless packet networks," in *Proc. IEEE International Conference on Image Processing*, pp. 132–135, Vancouver, BC, Canada, September 2000.

[7] J. Hua and Z. Xiong, "Optimal rate allocation in scalable joint source-channel coding for image transmission over CDMA networks," in *Proc. International Conference on Multimedia and Expo*, Baltimore, Md, USA, July 2003.

[8] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Joint product code optimization for scalable multimedia transmission over wireless channels," in *Proc. International Conference on Multimedia and Expo*, Lausanne, Switzerland, August 2002.

[9] M. Bystrom and J. W. Modestino, "Combined source-channel coding schemes for video transmission over an additive white Gaussian noise channel," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 880–890, 2000.

[10] M. Bystrom and J. W. Modestino, "Combined source-channel coding for transmission of H.263 coded video with trellis-coded modulation over a slow-fading Rician channel," in *Proc. IEEE International Symposium on Information Theory*, MIT, Cambridge, Mass, USA, August 1998.

[11] T. Chu and Z. Xiong, "Combined wavelet video coding and error control for Internet streaming and multicast," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 1, pp. 66–80, 2003.

[12] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," *ACM Computer Communication Review*, vol. 20, no. 4, pp. 200–208, 1990.

[13] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A transport protocol for real-time applications*, RFC 1889, January 1996.

[14] C. Bormann, L. Cline, G. Deisher, et al., *RTP payload format for the 1998 version of ITU-T Rec. H.263 video (H.263+)*, RFC 2429, October 1998.

[15] Y. Pei and J. W. Modestino, "A joint source-channel coding approach for packet video transport over wireless IP networks," in *Proc. 11th International Packet Video Workshop*, pp. 41–50, Kyongju, Korea, April 2001.

[16] S. Stein, "Fading channel issues in system engineering," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 2, pp. 68–89, 1987.

[17] P. Cherriman, C. H. Wong, and L. Hanzo, "Turbo- and BCH-coded wide-band burst-by-burst adaptive H.263-assisted wireless video telephony," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, no. 8, pp. 1355–1363, 2000.

[18] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Communications*, vol. 36, no. 4, pp. 389–400, 1988.

[19] Intel Corporation, "Video Codec Test Model, TMN8," June 1997, ftp://standard.pictel.com/video-site/h263plus/draft13.doc.

[20] D. Wu, T. Hou, and Y.-Q. Zhang, "Scalable video coding and transport over broadband wireless networks," *Proceedings of the IEEE*, vol. 89, no. 1, pp. 6–20, 2001.

[21] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the internet: Approaches and directions," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, 2001.

[22] R. Kurceren and J. W. Modestino, "A joint source-channel coding approach to scalable delivery of digital video over ATM networks," in *Proc. IEEE International Conference on Image Processing*, vol. 1, pp. 1599–1603, Vancouver, BC, Canada, September 2000.

[23] R. Floyd, B. Housel, and C. Tait, "Mobile Web access using eNetwork Web Express," *IEEE Personal Communications*, vol. 5, no. 5, pp. 47–52, 1998.

Yong Pei is currently a tenure-track Assistant Professor in the Computer Science and Engineering Department, Wright State University. Previously he was a Visiting Assistant Professor in the Electrical and Computer Engineering Department, University of Miami. He received his B.S. degree in electrical power engineering from Tsinghua University, Beijing, in 1996, and M.S. and Ph.D. degrees in electrical engineering from Rensselaer Polytechnic Institute in 1999 and 2002, respectively. His research interests include information theory, wireless communication systems and networks, and image/video compression and communications. He is a member of IEEE and Association for Computing Machinery (ACM).

James W. Modestino received the B.S. degree from Northeastern University, Boston, Mass, in 1962, and the M.S. degree from the University of Pennsylvania, Philadelphia, Pa, in 1964, both in electrical engineering. He also received the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, in 1968 and 1969, respectively. From 1970 to 1972, he was an Assistant Professor in the Department of Electrical Engineering, Northeastern University. In 1972, he joined Rensselaer Polytechnic Institute, Troy, NY, where until leaving in 2001 he was an Institute Professor in the Electrical, Computer and Systems Engineering Department and Director of the Center for Image Processing Research. In 2001 he joined the Department of Electrical and Computer Engineering at the University of Miami, Coral Gables, Fla, as the Victor E. Clarke Endowed Scholar, Professor and Chair. Dr. Modestino is a past member of the Board of Governors of the IEEE Information Theory Group. He is a past Associate Editor and book review editor for the IEEE Transactions on Information Theory. In 1984, he was corecipient of the Stephen O. Rice Prize Paper Award from the IEEE Communications Society and in 2000 he was corecipient of the Best Paper Award at the International Packet Video Conference.

EURASIP Journal on Applied Signal Processing 2004:2, 265–279

Scalable Video Transcaling for the Wireless Internet

Hayder Radha
Department of Electrical and Computer Engineering, Michigan State University, MI 48824-1226, USA
Email: radha@msu.edu

Mihaela van der Schaar
Department of Electrical and Computer Engineering, University of California, Davis, CA 95616-5294, USA
Email: mvanderschaar@ece.ucdavis.edu

Shirish Karande
Department of Electrical and Computer Engineering, Michigan State University, MI 48824-1226, USA
Email: karandes@msu.edu

Received 5 December 2002; Revised 28 July 2003

The rapid and unprecedented increase in the heterogeneity of multimedia networks and devices emphasizes the need for scalable and adaptive video solutions for both coding and transmission purposes. However, in general, there is an inherent trade-off between the level of scalability and the quality of scalable video streams. In other words, the higher the bandwidth variation, the lower the overall video quality of the scalable stream that is needed to support the desired bandwidth range. In this paper, we introduce the notion of wireless video transcaling (TS), which is a generalization of (nonscalable) transcoding. With TS, a scalable video stream, that covers a given bandwidth range, is mapped into one or more scalable video streams covering different bandwidth ranges. Our proposed TS framework exploits the fact that the level of heterogeneity changes at different points of the video distribution tree over wireless and mobile Internet networks. This provides the opportunity to improve the video quality by performing the appropriate TS process. We argue that an Internet/wireless network gateway represents a good candidate for performing TS. Moreover, we describe hierarchical TS (HTS), which provides a "transcaler" with the option of choosing among different levels of TS processes with different complexities. We illustrate the benefits of TS by considering the recently developed MPEG-4 fine granularity scalability (FGS) video coding. Extensive simulation results of video TS over bit rate ranges supported by emerging wireless LANs are presented.

Keywords and phrases: transcoding, FGS, scalable, video, transcaling, streaming.

1. INTRODUCTION

The level of heterogeneity in multimedia communications has been influenced significantly by new wireless LANs and mobile networks. In addition to supporting traditional web applications, these networks are emerging as important Internet video access systems. Meanwhile, both the Internet [1, 2, 3] and wireless networks are evolving to higher bit rate platforms with even larger amount of possible variations in bandwidth and other quality of services (QoS) parameters. For example, IEEE 802.11a and HiperLAN2 wireless LANs are supporting (physical layer) bit rates from 6 Mbps to 54 Mbps (see, e.g., [4, 5]). Within each of the supported bit rates, there are further variations in bandwidth due to the shared nature of the network and the heterogeneity of the devices and the quality of their physical connections. Moreover, wireless LANs are expected to provide higher bit rates than mobile networks (including third generation) [6]. In the meantime, it is expected that current wireless and mobile access networks (e.g., 2G and 2.5G mobile systems and sub-2 Mbps wireless LANs) will coexist with new generation systems for sometime to come. All of these developments indicate that the level of heterogeneity and the corresponding variation in available bandwidth could be increasing significantly as the Internet and wireless networks converge more and more into the future. In particular, if we consider the different wireless/mobile networks as a large multimedia heterogeneous access system for the Internet, we can appreciate the potential challenge in addressing the bandwidth variation over this system.

Many scalable video compression methods have been proposed and used extensively in addressing the bandwidth variation and heterogeneity aspects of the Internet and wireless networks (e.g., [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]). Examples of these include receiver-driven multicast multilayer coding, MPEG-4 fine granularity scalability (FGS)

compression, and H.263-based scalable methods. These and other similar approaches usually generate a base layer (BL) and one or more enhancement layers (ELs) to cover the desired bandwidth range. Consequently, these approaches can be used for multimedia unicast and multicast services over wireless Internet networks.

In general, the wider the bandwidth range[1] that needs to be covered by a scalable video stream, the lower the overall video quality is[2] [13]. With the aforementioned increase in heterogeneity over emerging wireless multimedia Internet protocol (IP) networks, there is a need for scalable video coding and distribution solutions that maintain good video quality while addressing the high level of anticipated bandwidth variation over these networks. One trivial solution is the generation of multiple streams that cover different bandwidth ranges. For example, a content provider, which is covering a major event, can generate one stream that covers 100–500 kbps, another that covers 500–1000 kbps and yet another stream to cover 1000–2000 kbps and so on. Although this solution may be viable under certain conditions, it is desirable from a content provider perspective to generate the fewest number of streams that covers the widest possible audience. Moreover, multicasting multiple scalable streams (each of which consists of multiple multicast sessions) is inefficient in terms of bandwidth utilization over the wired segment of the wireless IP network. (In the above example, a total bit rate of 3500 kbps is needed over a link transmitting the three streams, while only 2000 kbps of bandwidth is needed by a scalable stream that covers the same bandwidth range.)

In this paper, we propose a new approach for addressing the bandwidth variation issue over emerging wireless and mobile multimedia IP networks. We refer to this approach as transcaling (TS) since it represents a generalization of video transcoding. Video transcoding implies the mapping of a nonscalable video stream into another nonscalable stream coded at a bit rate *lower* than the first stream bit rate. With TS, one or more scalable streams covering different bandwidth ranges are derived from another scalable stream. While transcoding always degrades the video quality of the already-coded (nonscalable) video, a *transcaled* video could have a significantly better quality than the (original) scalable video stream prior to the TS operation. This represents a key difference between (nonscalable) transcoding and the proposed TS framework. TS can be supported at gateways between the wired Internet and wireless/mobile access networks (e.g., at a proxy server adjunct to an access point (AP) of a wireless LAN). We believe that this approach provides an efficient method for delivering good quality video over the high-bit rate wireless LANs while maintaining efficient utilization of the overall (wired/wireless) distribution network bandwidth. Therefore, different gateways of different wireless LANs and mobile networks can perform the desired TS operations that are suitable for their own local domains and the devices attached to them. This way, users of new higher-bandwidth LANs do not have to sacrifice in video quality due to coexisting with legacy wireless LANs or other low bit rate mobile networks. Similarly, powerful clients (e.g., laptops and PCs) can still receive high-quality video even if there are other low-bit-rate low-power devices that are being served by the same wireless/mobile network. Moreover, when combined with embedded video coding schemes and the basic tools of receiver-driven multicast, TS provides an efficient framework for video multicast over the wireless Internet.

In addition to introducing the notion of TS and describing how it can be used for unicast and multicast video services over wireless IP networks, we illustrate the level of quality improvement that TS can provide by presenting several video simulation results for a variety of TS cases. The remainder of the paper is organized as follows. Section 2 describes the wireless video TS framework with some focus on IP multicast applications. This section also highlights some of the key attributes and basic definitions of TS-based wireless systems and how they differ from traditional transcoding-based platforms. Section 3 describes hierarchical TS (HTS), which is a framework that enables transcalers to trade off video quality with complexity. HTS is described using a concrete example that is based on the MPEG-4 FGS video coding method. Then, two classes of TS are considered: full and partial. Section 4 described full TS for wireless LANs. Section 4 also shows simulation results of applying FTS on FGS streams and the level of video quality improvement one can gain by utilizing this approach. Section 5 complements Section 4 by describing partial TS and presenting results for performing PTS on the FGS temporal (FGST) coding method. Section 6 concludes the paper with a summary.

2. TRANSCALING-BASED MULTICAST (TSM) FOR VIDEO OVER THE WIRELESS INTERNET

A simple case of our proposed TS approach can be described within the context of receiver-driven layered multicast (RLM). Therefore, first, we briefly outline some of the basic characteristics of the RLM framework in order to highlight how this framework can be extended to our wireless video TS-based solution. Then, we describe some general features of a TS-based wireless Internet system.

RLM of video is based on generating a layered coded video bitstream that consists of multiple streams. The minimum quality stream is known as BL and the other streams are ELs [17]. These multiple video streams are mapped into a corresponding number of "multicast sessions." A receiver can subscribe to one (the BL stream) or more (BL plus one or more ELs) of these multicast sessions depending on the receiver's access bandwidth to the Internet. Receivers can subscribe to more multicast sessions or "unsubscribe" to some of the sessions in response to changes in the available

[1] A more formal definition of "bandwidth range" will be introduced later in the paper.

[2] This is particularly true for the scalable schemes that fall under the category of signal-to-noise ratio (SNR) scalability methods. These include the MPEG-2 and MPEG-4 SNR scalability methods, and the newly developed MPEG-4 FGS method.

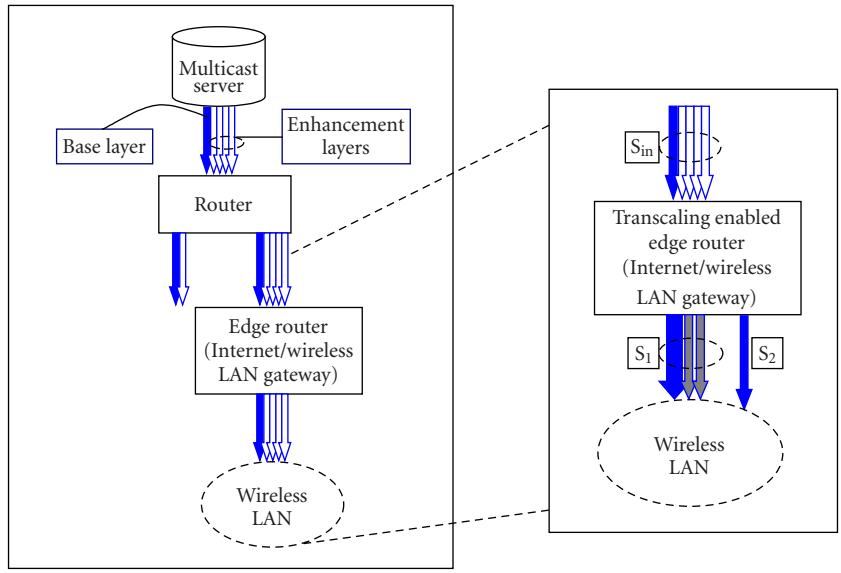


Figure 1: A simplified view of a wireless video TS platform within an RLM architecture.

bandwidth over time. The "subscribe" and "unsubscribe" requests generated by the receivers are forwarded upstream toward the multicast server by the different IP multicast-enabled routers between the receivers and the server. This approach results in an efficient distribution of video by utilizing minimal bandwidth resources over the multicast tree. The overall RLM framework can also be used for wireless IP devices that are capable of decoding the scalable content transmitted by an IP multicast server. The left picture of Figure 1 shows a simple example of an RLM-based system.

Similar to RLM, TS-based multicast (TSM) is driven by the receivers' available bandwidth and their corresponding requests for viewing scalable video content. However, there is a fundamental difference between the proposed TSM framework and traditional RLM. Under TSM, an edge router[3] with a TS capability (or a "transcaler") derives new scalable streams from the original stream. A derived scalable stream could have a BL and/or EL(s) that are different from the BL and/or ELs of the original scalable stream. The objective of the TS process is to improve the overall video quality by taking advantage of reduced uncertainties in the bandwidth variation at the edge nodes of the multicast tree.

For a wireless Internet multimedia service, an ideal location where TS can take place is at a gateway between the wired Internet and the wireless segment of the end-to-end network. The right picture of Figure 1 shows an example of a TSM system where a gateway node receives a layered video stream[4] with a BL bit rate $R_{\min\_in}$. The bit rate range covered by this layered set of streams is $R_{\text{range\_in}} = [R_{\min\_in}, R_{\max\_in}]$. The gateway transcales the input layered stream S_{in} into another scalable stream S_1. This new stream serves, for example, relatively high-bandwidth devices (e.g., laptops or PCs) over the wireless LAN. As shown in the figure, the new stream S_1 has a BL with a bit rate $R_{\min\_1}$ which is higher than the original BL bit rate: $R_{\min\_1} > R_{\min\_in}$. Consequently, in this example, the transcaler requires at least one additional piece of information, and that is the minimum bit rate $R_{\min\_1}$ needed to generate the new scalable video stream. This information can be determined, based on analyzing the wireless links of the different devices connected to the network.[5] By interacting with the access point, the gateway server can determine the bandwidth range needed for serving its devices. As illustrated by our simulations, this approach could improve the video quality delivered to higher-bit rate devices significantly.

2.1. Attributes of wireless video-transcaling-based systems

Here, we highlight the following attributes of the proposed wireless video TS framework.

(1) Supporting TS at edge nodes (wireless LANs' and mobile networks' gateways) preserves the ability of the local networks to serve low-bandwidth low-power devices (e.g., handheld devices). This is illustrated in Figure 1. In this example, in addition to generating the scalable stream S_1 (which has a BL bit rate that is

[3]The transcaling process does not necessarily take place in the edge router itself but rather in a proxy server (or a gateway) that is adjunct to the router.

[4]Here, a layered or "scalable" stream consists of multiple substreams.

[5]Determining the particular bit rate range over an underlying (wireless or wired) network is an important aspect of any adaptive multimedia solution, including TS. However, this aspect, which could include a variety of important topics and techniques such as congestion control, bandwidth estimation, and cross-layer communication and design, is beyond the scope of this paper.

higher than the bit rate of the input BL stream), the transcaler delivers the original BL stream to the low-bit rate devices.

(2) The TSM system (described above) falls under the umbrella of active networks[6] where, in this case, the transcaler provides network-based added value services [18]. Therefore, TSM can be viewed as a generalization of some recent work on active based networks with (nonscalable) video transcoding capabilities of MPEG streams.

(3) A wireless video transcaler can always fall back to using the original (lower-quality) scalable video. This "fall-back" feature represents a key attribute of TS that distinguishes it from nonscalable transcoding. The fall-back feature could be needed, for example, when the Internet wireless gateway (or whoever the transcaler happens to be) does not have enough processing power for performing the desired TS process(es). Therefore, and unlike (nonscalable) transcoding-based services, TS provides a scalable framework for delivering higher quality video. A more graceful TS framework (in terms of computational complexity) is also feasible as will be explained later in this paper.

(4) Although we have focused on describing our proposed wireless video TS approach in the context of multicast services, on-demand unicast applications can also take advantage of TS. For example, a wireless or mobile gateway may perform TS on a popular video clip that is anticipated to be viewed by many users on-demand. In this case, the gateway server has a better idea on the bandwidth variation that it (i.e., the server) has experienced in the past, and consequently, it generates the desired scalable stream through TS. This scalable stream can be stored locally for later viewing by the different devices served by the gateway.

(5) As illustrated by our simulation results, TS has its own limitations in improving the video quality over the whole desired bandwidth range. Nevertheless, the improvements that TS provides are significant enough to justify its merit over a subset of the desired bandwidth range. This aspect of TS will be explained further later in the paper.

(6) TS can be applied to any form of scalable streams (i.e., SNR, temporal, and/or spatial). In this paper, we will show examples of TS operations that are applied to SNR-scalable and hybrid SNR-temporal streams over bit rates that are applicable to new wireless LANs (e.g., 802.11). The level of improvement in video quality for both cases is also presented.

Before proceeding, it is important to introduce some basic definitions of TS. Here, we define two types of TS processes: down TS (DTS) and up TS (UTS).

[6]We should emphasize here that the area of active networks covers many aspects, and "added value services" is just one of these aspects.

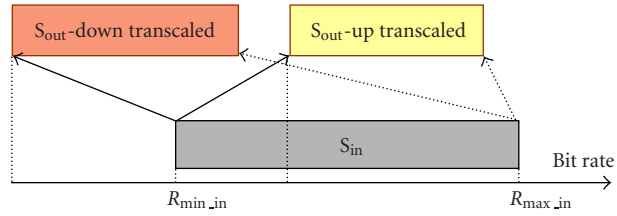


Figure 2: The distinction between DTS and UTS.

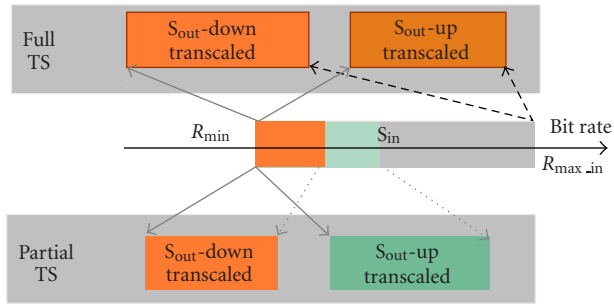


Figure 3: An example illustrating the different TS categories.

Let the original (input) scalable stream S_{in} of a transcaler covers a bandwidth range

$$R_{range\_in} = [R_{min\_in}, R_{max\_in}]. \quad (1)$$

And let a transcaled stream has a range

$$R_{range\_out} = [R_{min\_out}, R_{max\_out}]. \quad (2)$$

Then, DTS occurs when $R_{min\_out} < R_{min\_in}$, while UTS occurs when $R_{min\_in} < R_{min\_out} < R_{max\_in}$. The distinction between DTS and UTS is illustrated in Figure 2. DTS resembles traditional nonscalable transcoding in the sense that the bit rate of the output BL is lower than the bit rate of the input BL. Many researchers have studied this type of down conversion in the past.[7] However, up conversion has not received much attention (if any). Therefore, in the remainder of this paper, we will focus on UTS. (Unless otherwise mentioned, we will use UTS and TS interchangeably.)

Another important classification of TS is the distinction between FTS and PTS (see Figure 3). Our definition of FTS implies two things: (a) all of the input stream data (BL stream and EL stream) is used to perform the TS operation; and (b) *all* pictures of both BL and EL have been modified by TS. PTS is achieved if either of these two criteria is not met. Consequently, PTS provides a lower-complexity TS option that enables transcalers to trade off quality for complexity. Examples of both PTS and are covered in this paper.

[7]We should emphasize here, however, that we are not aware of any previous efforts of down converting a scalable stream into another scalable stream.

3. HIERARCHICAL TRANSCALING FOR THE WIRELESS INTERNET

After the above introduction to TS, its general features, potential benefits, and basic definitions, we now describe HTS for the wireless Internet. In order to provide a concrete example of HTS, we describe it in the context of the MPEG-4 FGS scalable video coding method. Hence, we start Section 3.1 with a very brief introduction to MPEG-4 FGS and its coding tools that have been developed in support of video streaming applications over the Internet and wireless networks.

3.1. The MPEG-4 FGS video coding method[8]

In order to meet the bandwidth variation requirements of the Internet and wireless networks, FGS encoding is designed to cover any desired bandwidth range while maintaining a very simple scalability structure [13]. As shown in Figure 4, the FGS structure consists of only two layers: a BL coded at a bit rate R_b and a single EL coded using a fine grained (or totally embedded) scheme to a maximum bit rate of R_e.

This structure provides a very efficient, yet simple, level of abstraction between the encoding and streaming processes. The encoder only needs to know the range of bandwidth $[R_{\min} = R_b, R_{\max} = R_e]$ over which it has to code the content, and it does not need to be aware of the particular bit rate the content will be streamed at. The streaming server on the other hand has a total flexibility in sending any desired portion of any EL frame (in parallel with the corresponding BL picture), without the need for performing complicated real-time rate control algorithms. This enables the server to handle a very large number of unicast streaming sessions and to adapt to their bandwidth variations in real time. On the receiver side, the FGS framework adds a small amount of complexity and memory requirements to any standard motion-compensation-based video decoder. As shown in Figure 4, the MPEG-4 FGS framework employs two encoders: one for the BL and the other for the EL. The BL is coded with the MPEG-4 motion compensation DCT-based video encoding method (nonscalable). The EL is coded using bit-plane-based embedded DCT coding.

FGS also supports temporal scalability (FGST) that allows for trade-offs between SNR and motion-smoothness at transmission time. Moreover, the FGS and FGST frames can be distributed using a single bitstream or two separate streams depending on the needs of the applications. Below, we will assume that MPEG-4 FGS/FGST video is transmitted using three separate streams: one for the BL, one for the SNR FGS frames, and the third one for the FGST frames.

For receiver-driven multicast applications (Figure 5), FGS provides a flexible framework for the encoding, streaming, and decoding processes. Identical to the unicast case, the encoder compresses the content using any desired range of bandwidth $[R_{\min} = R_b, R_{\max} = R_e]$. Therefore, the same compressed streams can be used for both unicast and multicast applications. At time of transmission, the multicast server partitions the FGS EL into any preferred number of "multicast channels" each of which can occupy any desired portion of the total bandwidth. At the decoder side, the receiver can "subscribe" to the "BL channel" and to any number of FGS EL channels that the receiver is capable of accessing (depending, e.g., on the receiver access bandwidth). It is important to note that regardless of the number of FGS EL channels that the receiver subscribes to, the decoder has to decode only a single EL.

The above advantages of the FGS framework are achieved while maintaining good coding efficiency results. However, similar to other scalable coding schemes, FGS's overall performance can degrade as the bandwidth range that an FGS stream covers increases.

3.2. Hierarchical transcaling of MPEG-4 FGS video streams

Examples of TS an MPEG-4 FGS stream are illustrated in Figure 6. Under the first example, the input FGS stream S_{in} is transcaled into another scalable stream S_1. In this case, the BL_{in} of S_{in} (with bit rate $R_{\min\_in}$) and a certain portion of the EL_{in} are used to generate a new base layer, BL_1. If R_{e1} represents the bit rate of the EL_{in} used to generate the new BL_1, then this new BL's bit rate $R_{\min\_1}$ satisfies the following:

$$R_{\min\_in} < R_{\min\_1} < R_{\min\_in} + R_{e1}. \tag{3}$$

Consequently, and based on the definition we adopted earlier for UTS and DTS, this example represents a UTS scenario. Furthermore, in this case, both the BL and EL of the input stream S_{in} has been modified. Consequently, this represents a FTS scenario. FTS can be implemented using cascaded decoder-encoder systems (as we will show in the simulation results section). This, in general could provide high-quality improvements at the expense of computational complexity at the gateway server.[9]

The residual signal between the original stream S_{in} and the new BL_1 stream is coded using FGS EL compression. Therefore, this is an example of TS an FGS stream with a bit rate range $R_{\text{range\_in}} = [R_{\min\_in}, R_{\max\_in}]$ to another FGS stream with a bit rate range $R_{\text{range\_1}} = [R_{\min\_1}, R_{\max\_1}]$. It is important to note that the maximum bit rate $R_{\max\_1}$ can be (and should be) selected to be smaller than the original maximum bit rate[10] $R_{\max\_in}$:

$$R_{\max\_1} < R_{\max\_in}. \tag{4}$$

[8]This brief subsection is mainly provided to make the paper self-contained. Readers who are familiar with the FGS framework can skip this subsection without affecting their understanding of the remainder of the paper.

[9]To reduce the complexity of FTS, one can reuse the motion vectors of the original FGS stream S_{in}. Reusing the same motion vectors, however, may not provide the best quality as has been shown in previous results for nonscalable TS.

[10]It is feasible that the *actual* maximum bit rate of the transcaled stream S_1 is higher than the maximum bit rate of the original input stream S_{in}. However, and as expected, this increase in bit rate does not provide any quality improvements as we will see in the simulation results. Consequently, it is important to truncate a transcaled stream at a bit rate $R_{\max\_1} < R_{\max\_in}$.

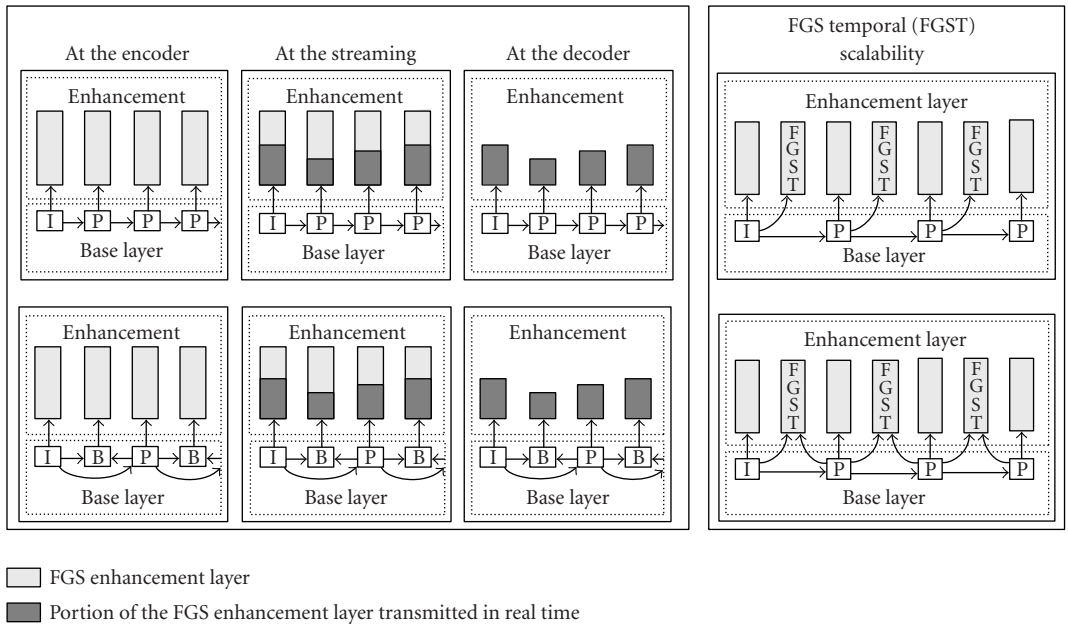


Figure 4: Examples of the MPEG-4 FGS and FGST scalability structures. Examples of the hybrid temporal SNR scalability structures are shown on the right-hand side of the figure. Both of bidirectional (lower right structure) and forward-prediction (top right figure) FGST picture types are supported by the MPEG-4 FGS/FGST standard.

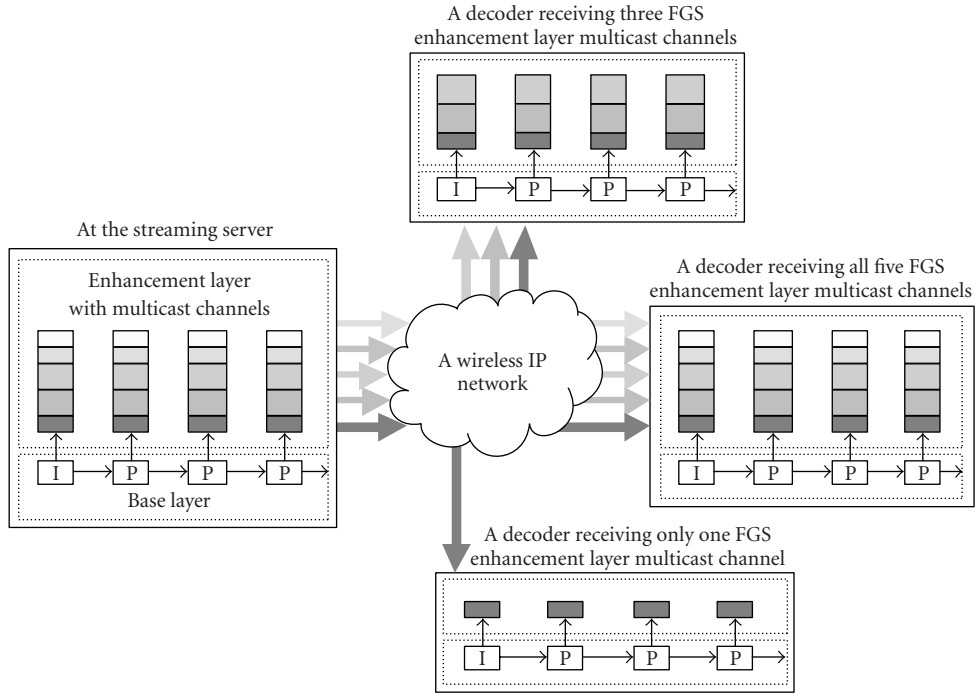


Figure 5: An example of video multicast using MPEG-4 FGS over a wireless IP network.

As we will see in the simulation section, the quality of the new stream S_1 at $R_{\max\_1}$ could still be higher than the quality of the original stream S_{in} at a higher bit rate $R \gg R_{\max\_1}$. Consequently, TS could enable a device which has a bandwidth $R \gg R_{\max\_1}$ to receive a better (or at least similar) quality video while saving some bandwidth. (This access bandwidth can be used, e.g., for other auxiliary or non-real-time applications.)

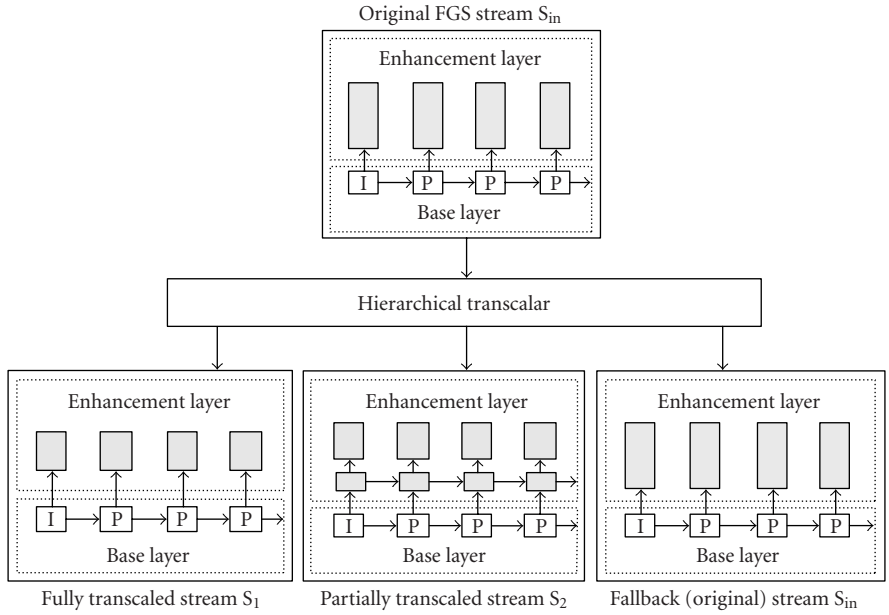


Figure 6: Examples of HTS of the MPEG-4 FGS scalability structure with an FTS option.

As mentioned above, under FTS, all pictures of both the BL and EL of the original FGS stream S_1 have been modified. Although the original motion vectors can be reused here, this process may be computationally complex for some gateway servers. In this case, the gateway can always fall back to the original FGS stream, and consequently, this provides some level of computational scalability.

Furthermore, FGS provides another option for TS. Here, the gateway server can transcales the EL only. This is achieved by (a) decoding a portion of the EL of one picture, and (b) using that decoded portion to predict the next picture of the EL, and so on. Therefore, in this case, the BL of the original FGS stream S_{in} is not modified and the computational complexity is reduced compared to FTS of the whole FGS stream (i.e., both BL and EL). Similar to the previous case, the motion vectors from the BL can be reused here for prediction within the EL to reduce the computational complexity significantly.

Figure 6 shows the three options described above for supporting HTS of FGS (SNR only) streams: FTS, PTS, and the fallback (no TS) option. Depending on the processing power available to the gateway, the system can select one of these options. The TS process with the higher complexity provides bigger improvements in video quality.

It is important to note that within each of the above TS options, one can identify further alternatives to achieve more graceful TS in terms computational complexity. For example, under each option, one may perform the desired TS on a fewer number of frames. This represents some form of temporal TS. Examples of this type of temporal TS and corresponding simulation results for wireless LANs bit rates are described in Section 5. Before proceeding, we show simulation results for FTS in the following section.

4. FULL TRANSCALING FOR HIGH-BIT-RATE WIRELESS LANS

In order to illustrate the level of video quality improvements that TS can provide for wireless Internet multimedia applications, in this section, we present some simulation results of FGS-based FTS.

We coded several video sequences using the draft standard of the MPEG-4 FGS encoding scheme. These sequences were then modified using the full transcaler architecture shown in Figure 7. The main objective for adopting the transcaler shown in the figure is to illustrate the potential of video TS and highlight some of its key advantages and limitations.[11]

The level of improvements achieved by TS depends on several factors. These factors include the type of video sequence that is being transcaled. For example, certain video sequences with a high degree of motion and scene changes are coded very efficiently with FGS [13]. Consequently, these sequences may not benefit significantly from TS. On the other end, sequences that contain detailed textures and exhibit a high degree of correlation among successive frames could benefit from TS significantly. Overall, most sequences gained visible quality improvements from TS.

Another key factor is the range of bit rates used for both the input and output streams. Therefore, we first need to

[11] Other elaborate architectures or algorithms can be used for performing FTS. However, these elaborate algorithms will bias some of our findings regarding the full potential of TS and its performance. Examples of these algorithms include refinement of motion vectors instead of a full recomputation of them; TS in the compressed DCT domain; and similar techniques.

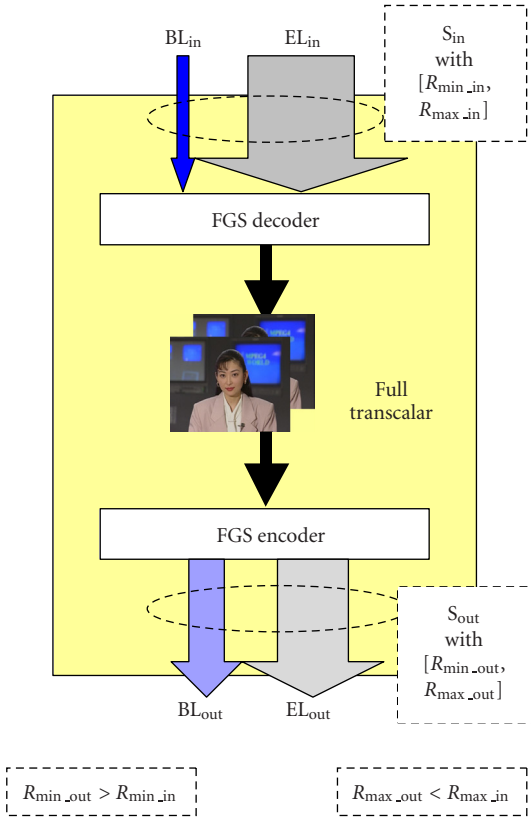


Figure 7: The full transcaler architecture used for generating the simulation results shown here.

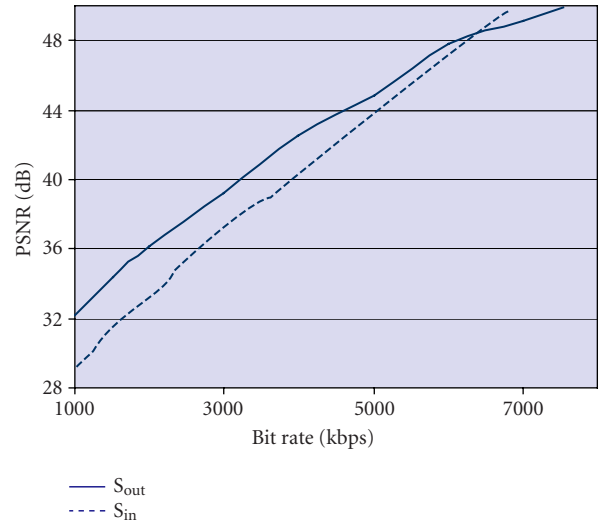


Figure 8: Performance of transcaling the Mobile sequence using an input stream S_{in} with a BL bit rate $R_{min\_in} = 250$ kbps into a stream with a BL $R_{min\_out} = 1$ Mbps.

decide on a reasonable set of bit rates that should be used in our simulations. As mentioned in the introduction, new wireless LANs (e.g., 802.11a or HiperLAN2) could have bit rates on the order of tens of Mbps (e.g., more than 50 Mbps). Although it is feasible that such high bit rates may be available to one or few devices at certain points in time, it is unreasonable to assume that a video sequence should be coded at such high bit rates. Moreover, in practice, most video sequences[12] can be coded very efficiently at bit rates below 10 Mbps. Consequently, the FGS sequences we coded were compressed at maximum bit rates (i.e., R_{max_in}) at around 6–8 Mbps. For the BL bit rate R_{min_in}, we used different values in the range of few hundreds kbps (e.g., between 100 and 500 kbps). Video parameters, which are suitable for the BL bit rates, were selected. All sequences were coded using CIF resolution and 10–15 frames/s.[13]

First, we present the results of TS an FGS stream ("Mobile") that has been coded originally with $R_{min\_in} = 250$ kbps and $R_{max\_in} = 8$ Mbps. The transcaler used a new BL bit rate $R_{min\_out} = 1$ Mbps. This example could represent a stream that was coded originally for transmission over lower bit rate systems (e.g., cable modem or legacy wireless LANs) and is being transcaled for transmission over new higher bit rate LANs. The peak SNR (PSNR) performance of the two streams as the functions of the bit rate is shown in Figure 8. (For more information about the MPEG-4 FGS encoding and decoding methods, the reader is referred to [13, 14].)

It is clear from the figure that there is a significant improvement in quality (close to 4 dB) in particular at bit rates close to the new BL rate of 1 Mbps. The figure also highlights that the improvements gained through TS are limited by the maximum performance of the input stream S_{in}. As the bit rate gets closer to the maximum input bit rate (8 Mbps), the performance of the transcaled stream saturates and gets closer (and eventually degrades below) the performance of the original FGS stream S_{in}. Nevertheless, for the majority of the desired bit rate range (i.e., above 1 Mbps), the performance of the transcaled stream is significantly higher. In order to appreciate the improvements gained through TS, we can compare the performance of the transcaled stream with that of an "ideal FGS" stream. Here, an ideal FGS stream is the one that has been generated from the original uncompressed sequence (i.e., not from a precompressed stream such as S_{in}). In this example, an ideal FGS stream is generated from the original sequence with a BL of 1 Mbps. Figure 9 shows the comparison between the transcaled stream and an ideal FGS stream over the range 1 to 4 Mbps. As shown in the figure, the performances of the transcaled and ideal streams are virtually identical over this range.

By increasing the range of bit rates that need to be covered by the transcaled stream, one would expect that its improvement in quality over the original FGS stream should get lower. Using the same original FGS (Mobile) stream coded with a BL bit rate of $R_{min\_in} = 250$ kbps, we transcaled this stream with a new BL bit rate $R_{min\_out} = 500$ kbps (i.e., lower than the 1 Mbps BL bit rate of the TS example described

[12] The exceptions to this statement are high-definition video sequences, which could benefit from bit rates around 20 Mbps.

[13] Our full transcaler used the exact same video parameters of the original video sequence (except bit rates) in order to avoid biasing the results.

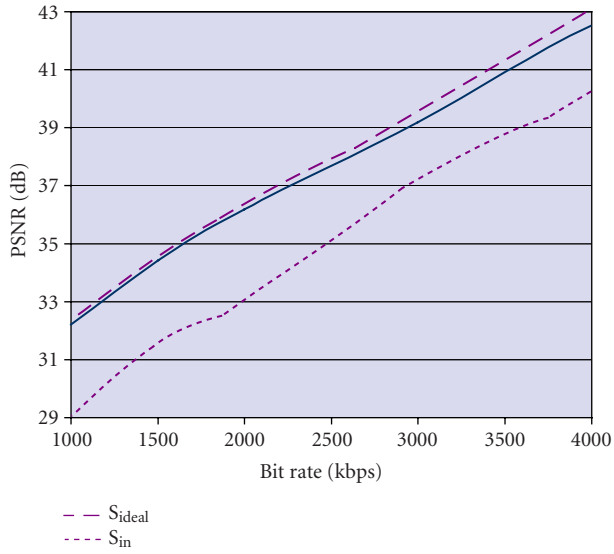


FIGURE 9: Comparing the performance of the Mobile transcaled stream (shown in Figure 8) with an ideal FGS stream. The performance of the transcaled stream is represented by the solid line.

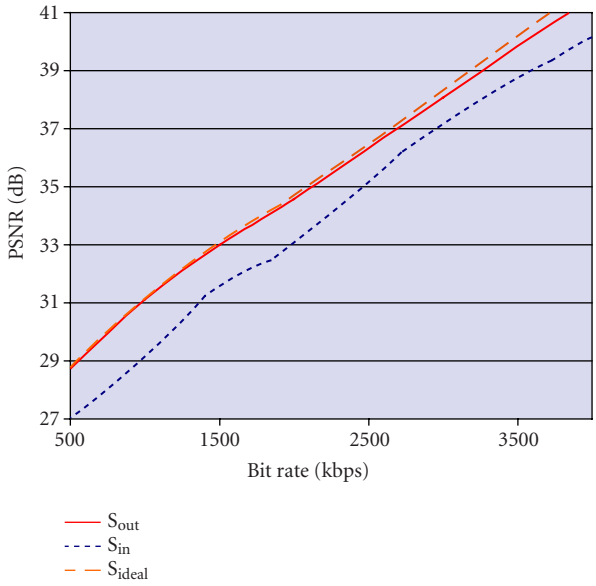


FIGURE 10: Performance of transcaling the Mobile sequence using an input stream S_{in} with a BL bit rate $R_{min\_in} = 250$ kbps into a stream with a BL $R_{min\_out} = 500$ kbps.

above). Figure 10 shows the PSNR performance of the input, transcaled, and ideal streams. Here, the PSNR improvement is as high as 2 dB around the new BL bit rate 500 kbps. These improvements are still significant (higher than 1 dB) for the majority of the bandwidth range. Similar to the previous example, we can see that the transcaled stream does saturates toward the performance of the input stream S_{in} at higher bit rates, and, overall, the performance of the transcaled stream is very close to the performance of the ideal FGS stream.

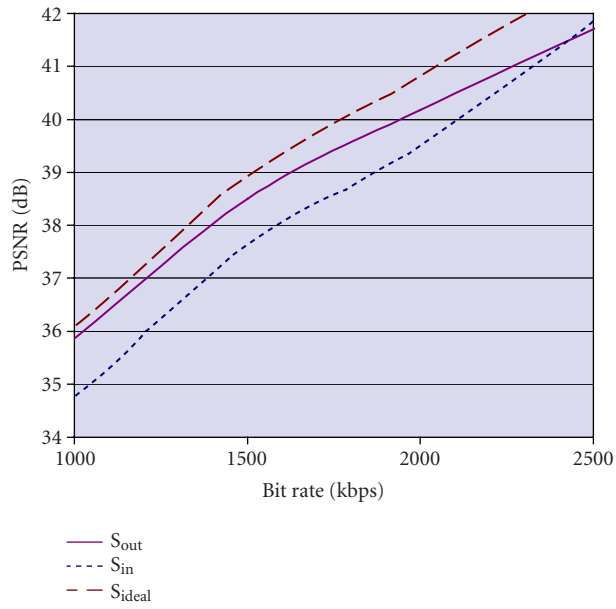


FIGURE 11: Performance of trascaling the Coastguard sequence using an input stream S_{in} with a BL bit rate $R_{min\_in} = 250$ kbps into a stream with a BL $R_{min\_out} = 1000$ kbps.

Therefore, TS provides rather significant improvements in video quality (around 1 dB and higher). The level of improvement is a function of the particular video sequences and the bit rate ranges of the input and output streams of the transcaler. For example, and as mentioned above, FGS provides different levels of performance depending on the type of video sequence [13]. Figure 11 illustrates the performance of TS the "Coastguard" MPEG-4 test sequence. The original MPEG-4 stream S_{in} has a BL bit rate $R_{min} = 250$ kbps and a maximum bit rate of 4 Mbps. Overall, FGS (without TS) provides a better quality scalable video for this sequence when compared with the performance of the previous sequence (Mobile). Moreover, the maximum bit rate used here for the original FGS stream ($R_{max\_in} = 4$ Mbps) is lower than the maximum bit rate used for the above Mobile sequence experiments. Both of these factors (i.e., a different sequence with a better FGS performance and a lower maximum bit rate for the original FGS stream S_{in}) led to the following: the level of improvements achieved in this case through TS is lower than the improvements we observed for the Mobile sequence. Nevertheless, significant gain in quality (more than 1 dB at 1 Mbps) can be noticed over a wide range over the transcaled bitstream. Moreover, we observe here the same "saturation-in-quality" behavior that characterized the previous Mobile sequence experiments. As the bit rate gets closer to the maximum rate R_{max_in}, the performance of the transcaled video approaches the performance of the original stream S_{in}.

The above results for TS were observed for a wide range of sequences and bit rates. So far, we have focused our attention on the performance of UTS, which we have referred to throughout this section simply by using the word TS. Now, we shift our focus to some simulation results for DTS.

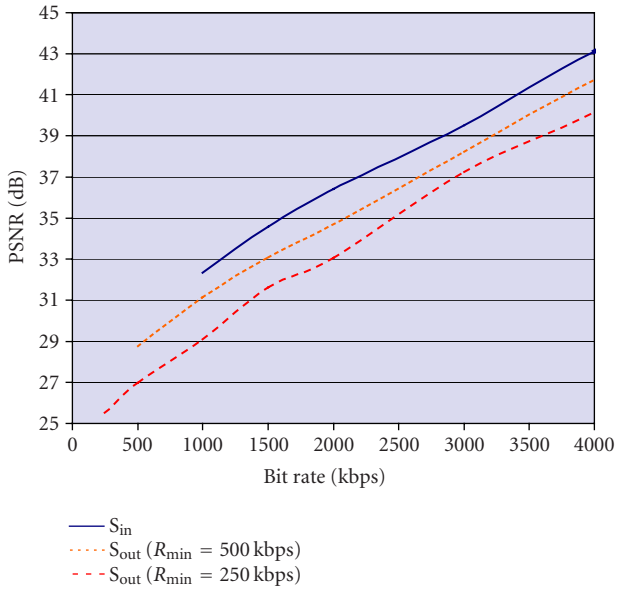


FIGURE 12: Performance of down transcaling the "Mobile" sequence using an input stream S_{in} with a BL bit rate R_{min_in} = 1 Mbps into two streams with BL R_{min_out} = 500 and 250 kbps.

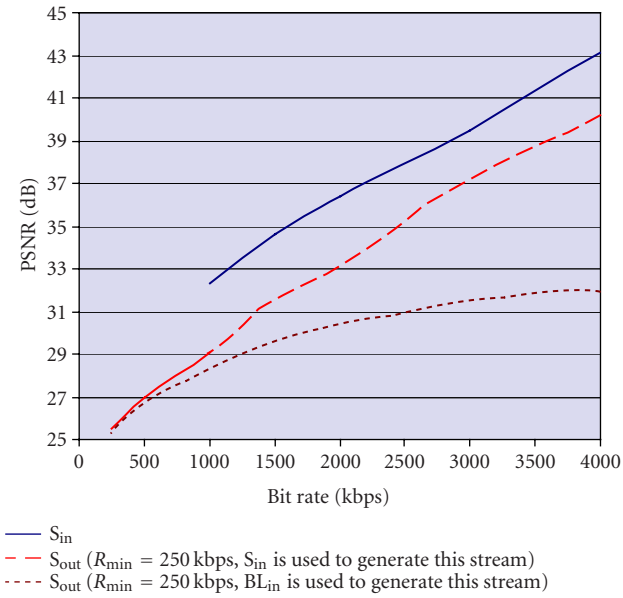


FIGURE 13: Performance of down transcaling the Mobile sequence using an input stream S_{in} with a BL bit rate R_{min_in} = 1 Mbps. Here, two DTS operations are compared, respectively, the whole input stream S_{in} (base + enhancement) is used, and only the BL_{in} of S_{in} is used to generate the down-transcaled stream. In both cases, the new DTS stream has a BL bit rate R_{min_out} = 250 kbps.

As explained above, DTS can be used to convert a scalable stream with a BL bit rate R_{min_in} into another stream with a smaller BL bit rate $R_{min\_out} < R_{min\_in}$. This scenario could be needed, for example, if (a) the transcaler gateway misestimates the range of bandwidth that it requires for its clients, (b) a new client appears over the wireless LAN, where this client has access bandwidth lower than the minimum bit rate (R_{min_in}) of the bitstream available to the transcaler; and/or (c) sudden local congestion over a wireless LAN is observed, and consequently, reducing the minimum bit rate needed. In this case, the transcaler has to generate a new scalable bitstream with a lower BL $R_{min\_out} < R_{min\_in}$. Below, we show some simulation results for DTS.

We employed the same full transcaler architecture shown in Figure 7. We also used the same Mobile sequence coded with MPEG-4 FGS and with a bit rate range R_{min_in} = 1 Mbps to R_{max_in} = 8 Mbps. Figure 12 illustrates the performance of the DTS operation for two bitstreams: one stream was generated by DTS the original FGS stream (with a BL of 1 Mbps) into a new scalable stream coded with a BL of R_{min_out} = 500 kbps. The second stream was generated using a new base layer R_{min_out} = 250 kbps. As expected, the DTS operation degrades the overall performance of the scalable stream.

It is important to note that, depending on the application (e.g., unicast versus multicast), the gateway server may utilize both the new generated (down-transcaled) stream and the original scalable stream for its different clients. In particular, since the quality of the original scalable stream S_{in} is higher than the quality of the down-transcaled stream S_{out} over the range $[R_{min\_in}, R_{max\_in}]$, then it should be clear that clients with access bandwidth that falls within this range can benefit from the higher quality (original) scalable stream S_{in}. On the other hand, clients with access bandwidth less than the original BL bit rate R_{min_in}, can only use the down-transcaled bitstream.

As mentioned in Section 2, DTS is similar to traditional transcoding, which converts a nonscalable bitstream into another nonscalable stream with a lower bit rate. However, DTS provides new options for performing the desired conversion that are not available with nonscalable transcoding. For example, under DTS, one may elect to use (a) both the BL and EL or (b) the BL only to perform the desired down conversion. This, for example, may be used to reduce the amount of processing power needed for the DTS operation. In this case, the transcaler has the option of performing only one decoding process (on the BL only versus decoding both the BL and EL). However, using the BL only to generate a new scalable stream limits the range of bandwidth that can be covered by the new scalable stream with an acceptable quality. To clarify this point, Figure 13 shows the performance of TS using (a) the entire input stream S_{in} (i.e., base plus enhancement) and (b) BL_{in} (only) of the input stream S_{in}. It is clear from the figure that the performance of the transcaled stream generated from BL_{in} saturates rather quickly and does not keep up with the performance of the other two streams. However, the performance of the second stream (b) is virtually identical over most of the range $[R_{min\_out} = 250\,\text{kbps}, R_{min\_in} = 500\,\text{kbps}]$. Consequently, if the transcaler is capable of using both the original stream S_{in} and the new transcaled stream S_{out} for transmission to its clients, then employing BL_{in} (only) to generate the new down-transcaled stream is a viable option.

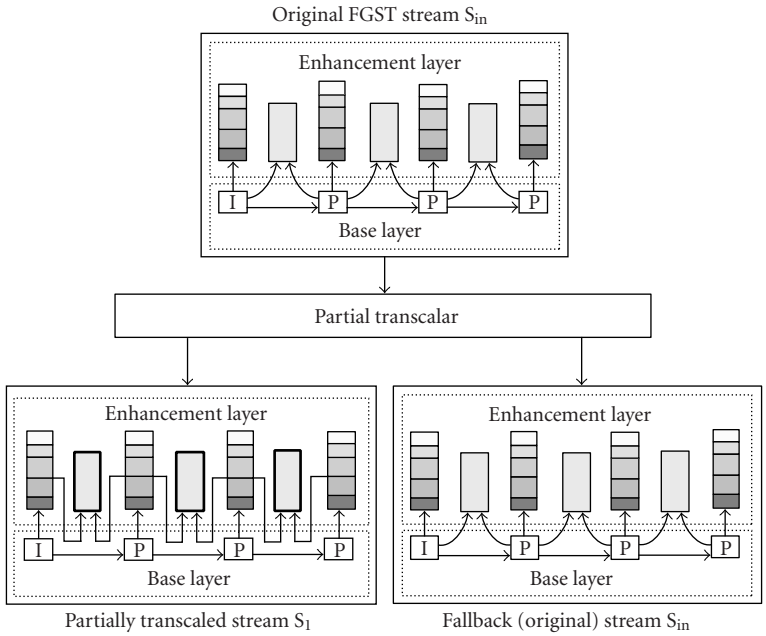


Figure 14: The proposed partial TS of the MPEG-4 FGST scalability structure. The FGST frames are the only part of the original scalable stream that is fully reencoded under the proposed partial TS scheme.

It is important to note that, in cases when the transcaler needs to employ a single scalable stream to transmit its content to its clients (e.g., multicast with a limited total bandwidth constraint), a transcaler can use the BL and any portion of the EL to generate the new down-transcaled scalable bitstream. The larger the portion of the EL used for DTS, the higher the quality of the resulting scalable video. Therefore, and since partial decoding of the EL represents some form of computational scalability, an FGS transcaler has the option of trading-off quality versus computational complexity when needed. It is important to note that this observation is applicable to both UTS and DTS.

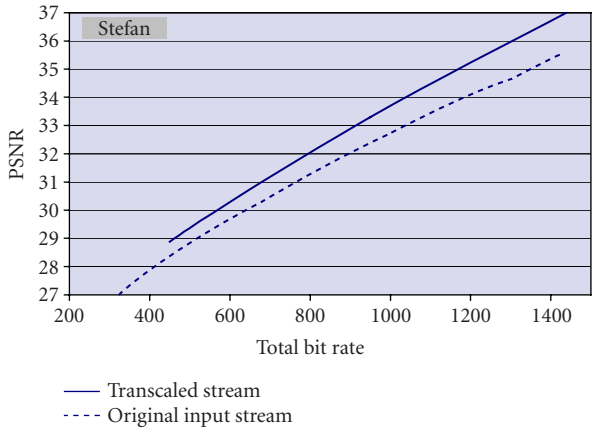
Finally, by examining Figure 13, one can infer the performance of a wide range of down-transcaled scalable streams. The lower-bound quality of these downscaled streams is represented by the quality of the bitstream generated from BL_{in} only (i.e., case (b) of S_{out}). Meanwhile, the upper-bound of the quality is represented by the downscaled stream (case (a) of S_{out}) generated by the full input stream S_{in}.

5. PARTIAL TRANSCALING FOR HIGH-BIT-RATE WIRELESS LANS

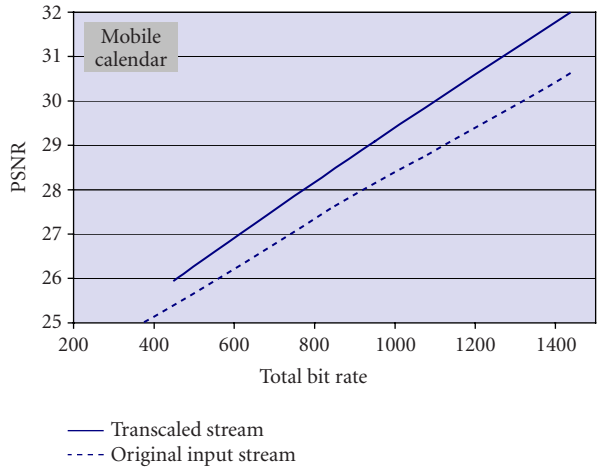
As described above, the MPEG-4 FGST framework supports SNR (regular FGS), temporal (FGST frames), and hybrid SNR-temporal scalabilities. At low bit rates (i.e., bit rates close to the BL bit rate), receivers can benefit from the standard SNR FGS scalability by streaming the BL and any desired portion of the SNR FGS enhancement-layer frames. As the available bandwidth increases, high-end receivers can benefit from both FGS and FGST pictures. It is important for these high-end receivers to experience higher-quality video when compared to the video quality of nontranscaled FGST streams. One of the reasons for the relatively high penalty in quality associated with the traditional FGST-based coding is that, at high bit rates, the FGST frames are predicted from low-quality (low bit rate) BL frames. Consequently, the resulting motion-compensated residual error is high, and thus a large number of bits are necessary for its compression.

In addition to improving the coding efficiency, it is crucial to develop a low complexity TS operation that provides the desirable improvements in quality. One approach for maintaining low complexity TS is to eliminate the need for reencoding the BL. Consequently, this eliminates the need for recomputing new motion vectors, which is the most costly part of a full transcaler that elects to perform this recomputation. Meanwhile, improvements can be achieved by using higher-quality (higher bit rate) SNR FGS pictures to predict the FGST frames. This reduces the entropy of the bidirectionally predicted FGST frames and, consequently leads to more coding efficiency or higher PSNR values. Examples of the input and output scalability structures of the proposed PTS scheme for FGST are depicted in Figure 14.

As shown in Figure 14, and similar to the full TS case, there are two options for supporting TS of FGST streams: the PTS option and the fallback (no TS) option. Depending on the processing power available to the gateway, the system can select one of these options. Every FGS SNR frame is shown with multiple layers, each of which can represent one of the bit planes of that frame. It is important to note that at higher bit rates, larger number of FGS SNR bit planes will be streamed, and consequently, these bit planes can be used to predict the FGST frames. Therefore, under an RLM framework, receivers that subscribe to the transcaled FGST stream

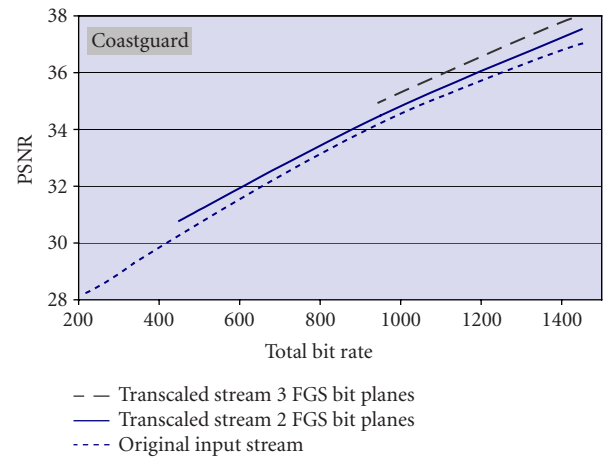


(a)

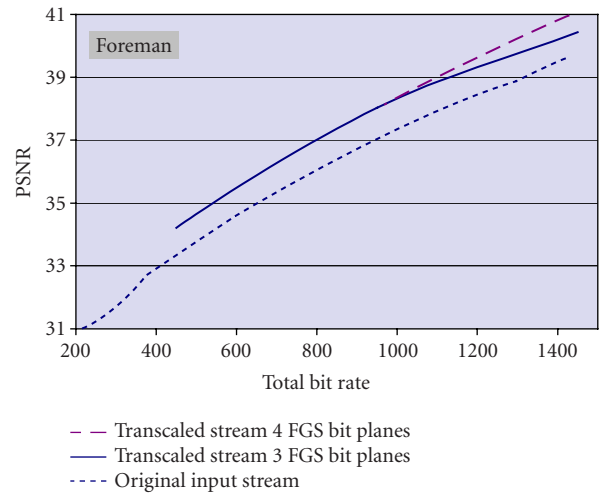


(b)

Figure 15: Performance of PTS of two sequences: Stefan and Mobile.



(a)



(b)

Figure 16: Performance of PTS of two sequences: Coastguard and Foreman.

should also subscribe to the appropriate number of FGS SNR bit planes.

Under the above-proposed PTS, the input FGST stream S_{in} is transcaled into another scalable stream S_1. In this case, BL_{in} of S_{in} (with bit rate R_{min_in}) and a certain portion of the EL_{in} are used as reference frames for an improved FGST performance. Therefore, this is an example of TS an FGST stream with a bit rate range $R_{range\_in} = [R_{min\_in}, R_{max\_in}]$ to another FGST stream with a bit rate range $R_{range\_1} = [R_{min\_1}, R_{max\_1}]$, where $R_{min\_in} < R_{min\_1}$. Consequently, and based on the definition we adopted earlier for UTS and DTS, this example represents a UTS scenario. Furthermore, in this case, only the FGST ELs of the input stream S_{in} has been modified. Consequently, this represents a PTS scenario. PTS can be implemented by using cascaded decoder-encoder systems for only part of the original scalable stream. It is important to note that, although we have a UTS scenario here, low-bandwidth receivers can still use the BL of the new transcaled stream, which is identical to the original BL. These receivers can also stream any desired portions of the FGS SNR frames. However, and as mentioned above, receivers that take advantage of the improved FGST frames have a new (higher) minimum bit rate stream ($R_{min\_1} > R_{min\_in}$) that is needed to decode the new FGST frames.

5.1. Simulation results for partial transcaling of FGST streams

In order to illustrate the level of video quality improvements that PTS can provide for wireless Internet applications, in this section, we present some simulation results of the FGST based PTS method described above. As in FTS experiments, we coded several video sequences using the MPEG-4 FGST scheme. These sequences were then modified using the partial transcaler scalability structure that employs a portion of the EL for FGST prediction as shown in Figure 14. We should emphasize here the following. (a) Unlike the FTS

results shown above, all the results presented in this section are based on reusing the same motion vectors that were originally computed by the BL encoder at the source. This is important for maintaining a low-complexity operation that can be realized in real time. (b) The FGS/FGST sequences we coded were compressed at maximum bit rates (i.e., $R_{\max\_in}$) lower than 2 Mbps. For the BL bit rate $R_{\min\_in}$, we used 50–100 kbps. Other video parameters, which are suitable for the BL bit rates, were selected. All sequences were coded using CIF resolution; however, and since the bit rate ranges are smaller than the FTS experiments, 10 frames/s were used in this case. The GOP size is 2-second long and M = 2 (i.e., one FGST bidirectionally predicted frame can be inserted between two I and P reference frames).

The PSNR performance of four well-known MPEG-4 streams: Foreman, Coastguard, Mobile, and Stefan have been simulated and measured for both original FGST (nontranscaled) and partially transcaled bitstreams over a wide range of bit rates.

Figure 15 shows the performance of the Stefan and Mobile (calendar) and compares the PSNR of the input nontranscaled stream with the partially transcaled streams' PSNR results. Both of these video sequences benefited from the PTS operation described above and gained as much as 1.5 dB in PSNR, in particular, at high bit rates. Three FGS bit planes were used (in addition to the BL) for predicting the FGST frames. Consequently, taking advantage of PTS requires that the receiver have enough bandwidth to receive the BL plus a minimum of three FGS bit planes. This explains why the gain in performance shown in Figure 15 begins at higher rates than the rate of the original BL bit rates (which are in the 50–100 kbps range as mentioned above).

As mentioned above, the level of gain obtained from the proposed PTS operation depends on the type of video sequence. Moreover, the number of FGS bit planes used for predicting the FGST frames influence the level of improvements in PSNR. Figure 16 shows the performance of the Coastguard and Foreman sequences. These sequences are usually coded more efficiently with FGS than the other two sequences shown above (Stefan and Mobile). Consequently, the improvements obtained by employing PTS on the Coastguard and Foreman sequences are less than the improvements observed in the above plots. Nevertheless, we are still able to gain about 1 dB in PSNR values at higher bit rates. Figure 16 also shows the impact of using different number of FGS bit planes from predicting the FGST frames. It is clear from both figures that, in general, larger number of bit planes provides higher gain in performance. However, it is important to note that this increase in PSNR gain (as the number of FGS bit planes used for prediction increases) could saturate as shown in the Foreman performance plots.

Furthermore, we should emphasize here that many of the video parameters used at the partial transcaler do not represent the best choice in a rate-distortion sense. For example, all of the results shown in this section are based on allocating the same number of bits to both the FGS and transcaled FGST frames. It is clear that a better rate allocation mechanism can be used. However, and as mentioned above, the main objective of this study is to illustrate the benefits and limitations of TS, in general, PTS, in particular, without the bias of different video parameters and related algorithms.

6. SUMMARY AND FUTURE WORK

In this paper, we introduced the notion of TS, which is a generalization of (nonscalable) transcoding. With TS, a scalable video stream, that covers a given bandwidth range, is mapped into one or more scalable video streams covering different bandwidth ranges. Our proposed TS framework exploits the fact that the level of heterogeneity changes at different points of the video distribution tree over wireless and mobile Internet networks. This provides the opportunity to improve the video quality by performing the appropriate TS process.

We argued that an Internet/wireless network gateway represents a good candidate for performing TS. Moreover, we described HTS, which provides a transcaler with the option of choosing among different levels of TS processes with different complexities. This enables transcalers to trade off video quality with computational complexity. We illustrated the benefits of FTS and PTS by considering the recently developed MPEG-4 FGS video coding.

Under FTS, we examined two forms: UTS (which we simply refer to as TS) and DTS. With UTS, significant improvements in video quality can be achieved as we illustrated in the simulation results section. Moreover, several scenarios for performing DTS were evaluated. Under PTS, we illustrated that a transcaler can still provide improved video quality (around 1 dB in improvements) while significantly reducing the high complexity associated with FTS. Consequently, we believe that the overall TS framework provides a viable option for the delivery of high-quality video over new and emerging high bit rate wireless LANs such 802.11a and 802.11b.

This paper has focused on the applied, practical, and proof-of-concept aspects of TS. Meanwhile, the proposed TS framework opens the door for many interesting research problems, some of which we are currently investigating. These problems include the following.

(1) A thorough analysis for an optimum rate-distortion (RD) approach for the TS of a wide range of video sequences is under way. This RD-based analysis, which is based on recent RD models for compressed scalable video [19], will provide robust estimation for the level of quality improvements that TS can provide for a given video sequence. Consequently, an RD-based analysis will provide an in-depth (or at least an educated) answer for: "when TS should be performed and on what type of sequences?"

(2) We are exploring new approaches for combining TS with other scalable video coding schemes such as 3D motion-compensated wavelets. Furthermore, TS in the context of cross-layer design of wireless networks is being evaluated [20, 21].

(3) Optimum networked TS that trades off complexity and quality in a distributed manner over a network of proxy video servers. Some aspects of this analysis

include distortion-complexity models for the different (full and partial) TS operations introduced in this paper. Moreover, other aspects of a networked TS framework will be investigated in the context of new and emerging paradigms such as overlay networks and video communications using path diversity (see, e.g., [22, 23, 24, 25, 26]).

ACKNOWLEDGMENTS

The authors would like to thank three anonymous reviewers who provided very constructive and valuable feedback on an earlier version of this paper. Many thanks to Professor Zixiang Xiong for his help and guidance throughout the review process. Parts of this work were presented at the ACM SIGMOBILE 2001 Workshop on Wireless Mobile Multimedia, Rome, Italy (in conjunction with MOBICOM 2001), the IEEE CAS 2001 Workshop on Wireless Communications and Networking, University of Notre Dame, and the Packet Video Workshop 2002.

REFERENCES

[1] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *ACM SIGCOMM '99*, pp. 263–274, Cambridge, Mass, USA, September 1999.
[2] V. Paxson, "End-to-end Internet packet dynamics," in *ACM SIGCOMM '97*, pp. 139–152, Cannes, France, September 1997.
[3] D. Loguinov and H. Radha, "Measurement study of low-bitrate Internet video streaming," in *ACM SIGCOMM Internet Measurement Workshop*, pp. 281–293, San Fransisco, Calif, USA, November 2001.
[4] B. Walke, N. Esseling, J. Habetha, et al., "IP over wireless mobile ATM – guaranteed wireless QoS by HiperLAN/2," *Proceedings of the IEEE*, vol. 89, no. 1, pp. 21–40, 2001.
[5] IEEE 802.11, "High Speed Physical Layer in the 5 GHz Band," 1999.
[6] R. Prasad, W. Mohr, and W. Konhäuser, Eds., *Third Generation Mobile Communication Systems*, Artech House, Boston, Mass, USA, 2000.
[7] M.-T. Sun and A. Reibmen, Eds., *Compressed Video over Networks*, Marcel Dekker, NY, USA, 2000.
[8] B. Girod and N. Farber, "Wireless video," in *Compressed Video over Networks*, Marcel Dekker, NY, USA, 2000.
[9] H. Radha, C. Ngu, T. Sato, and M. Balakrishnan, "Multimedia over wireless," in *Advances in Multimedia: Systems, Standards, and Networks*, Marcel Dekker, NY, USA, March 2000.
[10] M. R. Civanlar, "Internet video," in *Advances in Multimedia: Systems, Standards, and Networks*, Marcel Dekker, NY, USA, March 2000.
[11] W. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 172–186, 1999.
[12] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable Internet video using MPEG-4," *Signal Processing: Image Communication*, vol. 15, no. 1-2, pp. 95–126, 1999.
[13] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 FGS video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, 2001.
[14] ISO/IEC 14496-2, "Information Technology – Coding of Audio-Visual Objects – part 2: Visual," International Standard, ISO/IEC JTC 1/SC 29/WG 11, March 2000.
[15] D. Wu, Y. T. Hou, and Y.-Q. Zhang, "Scalable video coding and transport over broadband wireless networks," *Proceedings of the IEEE*, vol. 89, no. 1, pp. 6–20, 2001.
[16] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 6, pp. 983–1001, 1997.
[17] S. McCanne, V. Jackobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. Special Interest Group on Data Communications*, pp. 117–130, Standford, Calif, USA, August 1996.
[18] K. L. Calvert, A. T. Campbell, A. A. Lazar, D. Wetherall, and R. Yavatkar eds., "Special issue on active and programmable networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 3, 2001.
[19] M. Dai, D. Loguinov, and H. Radha, "Statistical analysis and distortion modeling of MPEG-4 FGS," in *IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003.
[20] S. A. Khayam, S. S. Karande, M. Krappel, and H. Radha, "Cross-layer protocol design for real-time multimedia applications over 802.11b networks," in *IEEE International Conference on Multimedia and Expo*, Baltimore, Md, USA, July 2003.
[21] S. A. Khayam, S. S. Karande, H. Radha, and D. Loguinov, "Performance analysis and modeling of errors and losses over 802.11b LANs for high-bitrate real-time multimedia," *Signal Processing: Image Communication*, vol. 18, no. 7, pp. 575–595, 2003.
[22] "Session on overlay networks," in *Proc. Special Interest Group on Data Communications*, Pittsburgh, Pa, USA, August 2002.
[23] D. Towsley, C. Diot, B. N. Levine, and L. Rizzo eds., "Special issue on network support for multicast communications," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, 2002.
[24] "Session on overlay routing and multicast," in *INFOCOM 2003*, San Francisco, Calif, USA, April 2003.
[25] J. Taal, I. Haratcherev, K. Langendoen, and R. Lagendijk, "Special session on networked video (SS-LI)," in *IEEE International Conference on Multimedia and Expo*, Baltimore, Md, USA, July 2003.
[26] "Lecture session on multimedia streaming: MCN-L5," in *IEEE International Conference on Multimedia and Expo*, Baltimore, Md, USA, July 2003.

Hayder Radha received his B.S. degree (with honors) from Michigan State University (MSU) in 1984, his M.S. degree from Purdue University in 1986, and his Ph.M. and Ph.D. degrees from Columbia University in 1991 and 1993, all in electrical engineering. He joined MSU in 2000 as an Associate Professor in the Department of Electrical and Computer Engineering. Between 1996 and 2000, Dr. Radha was with Philips Research, USA, first as a Principal Member of the research staff and then as a Consulting Scientist. In 1997, Dr. Radha initiated the Internet video project and led a team of researchers working on scalable video coding, networking, and streaming algorithms. Prior to working at Philips Research, Dr. Radha was a member of technical staff at Bell Labs, where he worked between 1986 and 1996 in the areas of digital communications, signal/image processing, and broadband multimedia. Dr. Radha is the recipient of the Bell Labs Distinguished Member of Technical Staff Award and the Withrow Junior Distinguished Scholar Award. He was appointed as a Philips

Research Fellow in 2000. His research interests include image and video coding, wireless communications, and multimedia networking. He has 25 patents (granted or pending) in these areas.

Mihaela van der Schaar is currently an Assistant Professor in the Electrical and Computer Engineering Department at the University of California, Davis. She received her Ph.D. degree in electrical engineering from Eindhoven University of Technology, the Netherlands. Between 1996 and June 2003, she was a Senior Member Research Staff at Philips Research in the Netherlands and USA. In 1998, she worked in the Wireless Communications and Networking Department. From January to September 2003, she was also an Adjunct Assistant Professor at Columbia University. In 1999, she become an active participant to the MPEG-4 standard, contributing to the scalable video coding activities. She is currently chairing the MPEG Ad-hoc group on Scalable Video Coding, and is also cochairing the Ad-hoc group on Multimedia Test Bed. Her research interests include multimedia coding, processing, networking, and architectures. She has authored more than 70 book chapters, and conference and journal papers and holds 9 patents and several more pending. She was also elected as a member of the Technical Committee on Multimedia Signal Processing of the IEEE Signal Processing Society and is an Associate Editor of IEEE Transactions on Multimedia and an Associate Editor of Optical Engineering.

Shirish Karande is currently pursuing his Ph.D. at Michigan State University (MSU). He received his B.E. degree in electronics and telecommunications from University of Pune in 2000, and his M.S. degree in electrical engineering from MSU in 2003. He was the recipient of the Government of India National Talent Search (NTS) Merit Scholarship from 1994 to 2000. His research interests include scalable source coding, channel coding, and wireless networking.

EURASIP Journal on Applied Signal Processing 2004:2, 280–289

Effective Quality-of-Service Renegotiating Schemes for Streaming Video

Hwangjun Song
School of Electrical Engineering, Hongik University, 72-1 Sangsu-dong, Mapo-gu, Seoul 121-791, Korea
Email: hwangjun@wow.hongik.ac.kr

Dai-Boong Lee
School of Electrical Engineering, Hongik University, 72-1 Sangsu-dong, Mapo-gu, Seoul 121-791, Korea
Email: neferian@hotmail.com

Received 13 November 2002; Revised 25 September 2003

Effective quality-of-service renegotiating schemes for streaming video is presented. The conventional network supporting quality of service generally allows a negotiation at a call setup. However, it is not efficient for the video application since the compressed video traffic is statistically nonstationary. Thus, we consider the network supporting quality-of-service renegotiations during the data transmission and study effective quality-of-service renegotiating schemes for streaming video. The token bucket model, whose parameters are token filling rate and token bucket size, is adopted for the video traffic model. The renegotiating time instants and the parameters are determined by analyzing the statistical information of compressed video traffic. In this paper, two renegotiating approaches, that is, fixed renegotiating interval case and variable renegotiating interval case, are examined. Finally, the experimental results are provided to show the performance of the proposed schemes.

Keywords and phrases: streaming video, quality-of-service, token bucket, renegotiation.

1. INTRODUCTION

In recent years, the demands and interests in networked video have been growing very fast. Various video applications are already available over the network, and the video data is expected to be one of the most significant components among the traffics over the network in the near future. However, it is not a simple problem to transmit video traffics efficiently through the network because the video requires a large amount of data compared to other multimedia. To reduce the amount of data, it is indispensable to employ effective video compression algorithms. So far, digital video coding techniques have advanced rapidly. International standards such as MPEG-1, MPEG-2 [1], MPEG-4 [2], H.261 [3], H.263/+/++ [4], H.26L, and H.264 have been established or are under development to accommodate different needs by ISO/IEC and ITU-T, respectively. The compressed video data is generally of variable bit rate due to the generic characteristics of entropy coder and scene change inconsistent motion change of the underlying video. Furthermore, video data is time constrained. These facts make the problem more challenging. By the way, constant bit rate video traffic can be generated by controlling the quantization parameters and it is much easier to handle over the network, but the quality of the decoded video may be seriously degraded.

In general, suitable communications between the network and the sender end can increase the network utilization and enhance video quality at the receiver end simultaneously [5]. Generally speaking, the variability of compressed video traffics consists of two components: short-term variability (or high-frequency variability) and long-term variability (or low-frequency variability). Buffering is only effective in reducing losses caused by variability in the high-frequency domain, and is not effective in handling variability in the low-frequency domain [6]. Recently, some QoS (quality-of-service) renegotiating approaches have been proposed to handle the nonstationary video traffics efficiently over the network [7, 8, 9, 10, 11, 12], while the conventional QoS providing network negotiates QoS parameters only once at a call setup. For example, RCBR (renegotiated constant bit rate) [7, 8] is a simple but quite effective approach to support the QoS renegotiations. RCBR network allows the sender to renegotiate the bandwidth during the data transmission. Actually, the bandwidth renegotiations can be interpreted as a compromise of ABR (available bit rate) and VBR (variable bit rate). Over network supporting bandwidth renegotiations, how to determine the renegotiation instants and the required bandwidth is studied in [9, 10, 11, 12, 13]. In [11], Zhang and Knightly proposed the RED-VBR (renegotiated deterministic variable bit rate) service model to support VBR video that

uses a traffic model called D-BIND (deterministic bounding interval-length dependent). Salehi et al. proposed the shortest path algorithm to reduce the number of renegotiations and the bandwidth fluctuation in [12]. In our previous work [10], we studied adaptive rate-control algorithms to pursue an effective trade-off between temporal and spatial qualities for streaming video and interactive video applications over RCBR network.

However, only bandwidth renegotiation is sometimes not sufficient to efficiently support the nonstationary video traffics and improve the network utilization. (The higher network utilization means that the better services are provided to users and/or more users are supported with the same network resources.) Generally speaking, more network resources are required for the media delivery as its traffic becomes more burst although the long-term average bandwidth is the same. Thus, we need more flexible QoS renegotiating approaches for streaming videos to improve network utilization and enhance video quality at the receivers end. In this paper, we consider not only channel bandwidth but also the burstiness of the traffic. To handle the problem, token bucket is adopted for the traffic model, and its parameters are estimated based on the statistical characteristics of compressed video traffic during the data transmission. This paper is organized as follows: a brief review of traffic models is introduced in Section 2; effective QoS renegotiating schemes are proposed in Section 3; experimental results are provided in Section 4 to show the superior performance of the proposed schemes; and finally, concluding remarks are presented in Section 5.

2. TRAFFIC MODEL

So far, various traffic models have been proposed for efficient network resource management such as policing, resource reservation, rate shaping, and so forth. For example, leaky bucket model [14], double leaky bucket model [15], token bucket model [16, 17], and so forth. As mentioned earlier, the token bucket model is adopted in this paper, which is one of the most popular traffic models and widely employed for IntServ protocol [18]. In the token bucket model, each packet can be transmitted through the network with one token only when tokens are available at the token buffer. The tokens are generally provided by network with a fixed rate. When the token buffer is empty, the packet must wait for a token in the smoothing buffer. On the other hand, the new arriving tokens are dropped when the token bucket is full. It means the waste of network resource. The token bucket model can be characterized by two parameters: token filling rate and token bucket size. The token filling rate and the token bucket size are related to the average channel bandwidth and the burstiness of the underlying video traffic, respectively. In general, more burst traffic needs a larger token bucket size, and complex token model has one more parameter than simple bucket model, that is, it can be characterized by the token filling rate, token bucket size, and peak rate. Their performance comparison can be found in [19].

An overview of simple bucket model is shown in Figure 1.

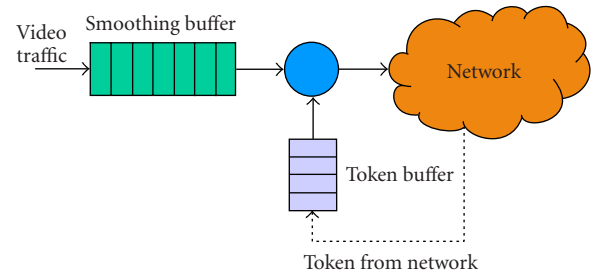


Figure 1: Overview of the simple token bucket model.

The token bucket is thought to be located in either the user side or the network side. The network needs the token bucket to policy the incoming traffics while the user requires the token bucket to generate the video traffic according to the predetermined specification. Smoothing buffer is also an important factor to determine the video traffic characteristics, which relates to packet loss rate and time delay. Since the smoothing buffer size is practically finite, buffer management algorithm is needed to minimize the degradation of video quality caused by buffer overflow. In this paper, the following buffer management is employed: B-, P-, and I-frames are discarded in sequence when smoothing buffer overflows. It is determined by how much the quality of the decoded video may be degraded when a frame is lost. When the I-frame is dropped, the whole GOPs (group of pictures) cannot be decoded since the I-frame is referenced for the following P-frames and B-frames. When the P-frame is dropped, the following frames in the GOPs disappear. However, only one frame is missing when the B-frame is dropped since the other frames do not reference it. To more improve the video quality, network needs to classify the incoming packets and consider the error corruption in the whole sequences caused by a specific packet loss [20, 21]. However, it is a big burden to network because of a large amount of computation. In this paper, we consider the renegotiations of token bucket parameters during data transmission as a solution to improve network utilization and enhance video quality at the receiver end.

3. PROPOSED TOKEN BUCKET PARAMETER ESTIMATING SCHEMES

Over the network supporting QoS renegotiations, the sender has to determine when QoS renegotiation is required and what QoS is needed for. Note that, in general, more renegotiations can increase the network utilization; however, they may cause larger signaling overhead. We assume that the compressed data for each frame is divided into fixed size packets, and thus the number of packets (N_i) for the ith frame is calculated by

$$N_i = \left\lfloor \frac{B_i}{P_{\max}} \right\rfloor, \tag{1}$$

where $\lfloor x \rfloor$ indicates the smallest integer that is greater than x, B_i is the amount of bits for the compressed ith frame,

and $P_{\max}$ is the packet size. Under the assumption that the video stream is accepted by call admission control, we focus on only the QoS renegotiating process in this paper. In many cases, the compressed data may not be divided into the fixed size packets for the robust transmission. However, the above assumption is still reasonable if packets are assumed to consume the different number of tokens according to their size.

We examine two approaches for the QoS renegotiation: fixed renegotiating interval approach and variable renegotiating interval approach. Renegotiations are tried periodically in the fixed renegotiating interval case while they are tried only when required in the variable renegotiating interval case. It is expected that variable renegotiating interval approach can avoid unnecessary renegotiations and unsuitable renegotiating instants with higher computational complexity. In each renegotiating interval, we estimate the required token bucket parameters based on the statistical information of video traffic. That is, token filling rate and token bucket size are determined by the mean and the standard deviation of number of packets, respectively.

3.1. Fixed renegotiating interval case

First of all, the statistical information, mean and standard deviation of the underlying video traffic, is calculated in the reference window, and then the token bucket model parameters, token filling rate, and token bucket size are estimated to keep the packet loss rate in the tolerable range. Then, the whole time interval of the underlying video are divided into time intervals with the same size, and the mean and the standard deviation are calculated in each interval. Based on the information, the required token bucket model parameters in the arbitrary renegotiating interval are determined. The above processes can be summarized as follows: renegotiations are tried at every interval with these parameters:

$$R_i = \left(1 + \alpha \frac{m_i - M_{\text{ref}}}{M_{\text{ref}}}\right) \cdot R_{\text{ref}}, \tag{2}$$

$$Q_i = \left(1 + \beta \frac{\sigma_i - \sigma_{\text{ref}}}{\sigma_{\text{ref}}}\right) \cdot Q_{\text{ref}}, \tag{3}$$

where M_{ref} and m_i are the mean values of numbers of packets for each frame in the reference window; the ith renegotiating interval, respectively, σ_{ref} and σ_i are the standard deviations of numbers of packets for each frame in the reference window; the ith renegotiating interval, respectively, α and β are the weighting factors; R_i and Q_i are the token filling rate and the token bucket size in the ith renegotiating interval, respectively; and R_{ref} and Q_{ref} are the token filling rate and the token bucket size in the reference window, respectively. We assume that the number of packets for a frame in the reference window is Gaussian distributed for the simplicity, and then R_{ref} and Q_{ref} are determined by

$$\begin{aligned} R_{\text{ref}} &= \frac{\sum_{i=1}^{F_{\text{ref}}} N_i}{F_{\text{ref}}}, \\ Q_{\text{ref}} &= \sigma_{\text{ref}} \cdot I + M_{\text{ref}}, \end{aligned} \tag{4}$$

where F_{ref} is the number of frames in the reference window and I satisfies the following equation:

$$\Pr(X > I) \le p, \tag{5}$$

where X is a Gaussian random variable with zero mean and unit standard deviation, and p is the tolerable packet loss probability.

3.2. Variable renegotiating interval case

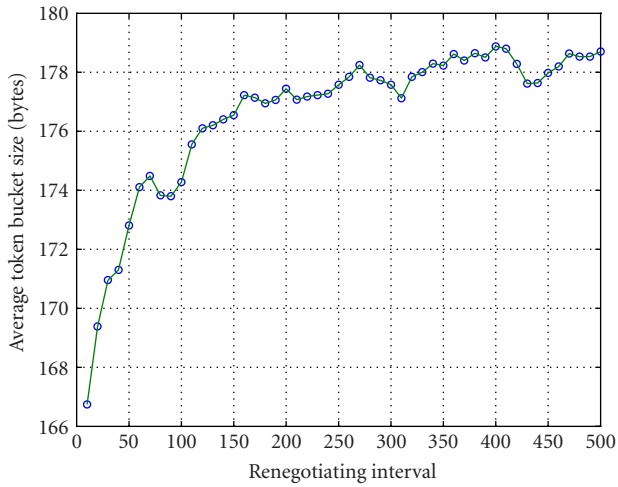
When the fixed renegotiating interval approach is tested, undesirable phenomena are sometimes observed. That is, the average token bucket size, token drop rate, and packet loss rate locally fluctuate as shown in Figures 2 and 3 even though their general trends globally decrease as the average renegotiating interval becomes small. One of the reasons is that the fixed renegotiating interval can make the inappropriate interval segmentation. To solve this problem, we consider a variable renegotiating interval approach. Now, we define the basic renegotiating interval unit consisting of several GOPs and address how to determine the renegotiating instants by using the basic unit. As shown in Figures 2 and 3 (the fixed renegotiating interval case), the graphs of average token bucket size, token drop rate, and packet loss rate look very similar. Thus, one of them can be used as a measure for the determination of renegotiating instants. In this paper, packet loss rate is employed. First, we calculate the packet loss rate in the current window, that is, the time interval since the latest renegotiation, and compute the new packet loss rate when the next basic renegotiating interval is included in the window. Second, we determine whether the next basic renegotiating interval is included or not in the window based on the difference between the two packet loss rates. It can be summarized as follows. If

$$\frac{\text{PLR}_{\text{next}}}{\text{PLR}_{\text{cur}}} > 1 + T(\mu, n), \tag{6}$$

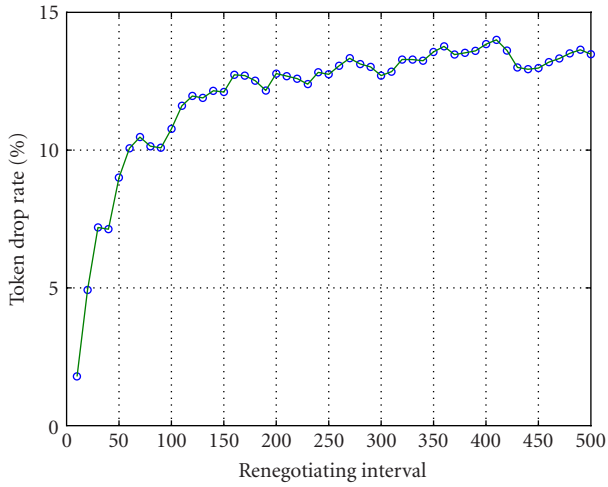
then the next basic interval is not included in the window. Otherwise, the next basic interval is included in the window. Where PLR_{cur} is the packet loss rate in the current window, PLR_{next} is the packet loss rate when the next basic renegotiating interval is included in the current window, n is the number of the minimum renegotiating intervals in the current window, μ is a variable determining the number of renegotiations, and $T(\mu, n)$ is a threshold function which must take into account the fact that the effect of the next basic renegotiating interval on PLP_{next} decreases as the window size increases. In this paper, $T(\mu, n)$ is simply defined by

$$T(\mu, n) = \frac{\mu}{100 \cdot n}. \tag{7}$$

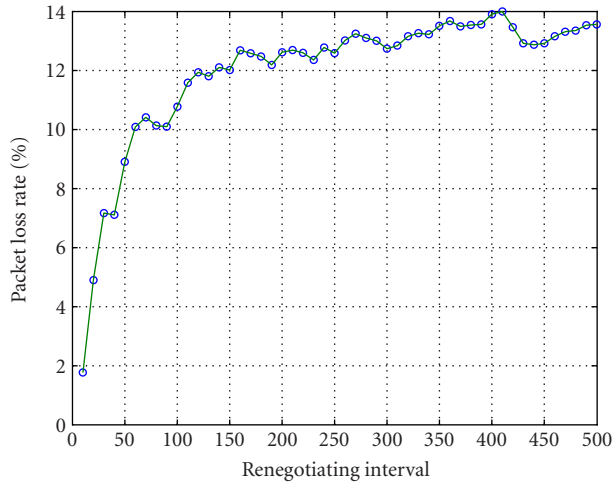
If the renegotiating instant is determined by the above process, the token bucket model parameters for the current interval are estimated by the same method ((2) and (3)) of the fixed renegotiating interval case. Basically, the length of the basic renegotiating interval unit is related to the network utilization and the computational complexity. As the length becomes smaller, network utilization can be improved while the required computational complexity increases.



(a)

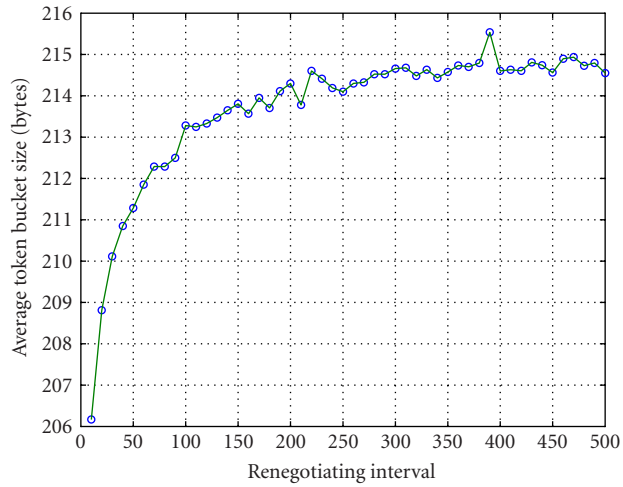


(b)

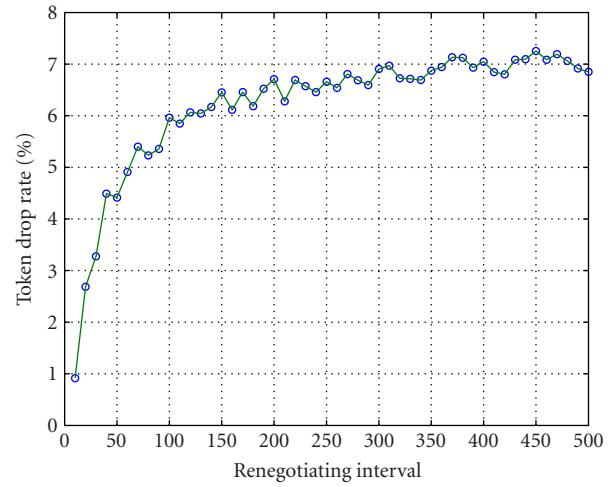


(c)

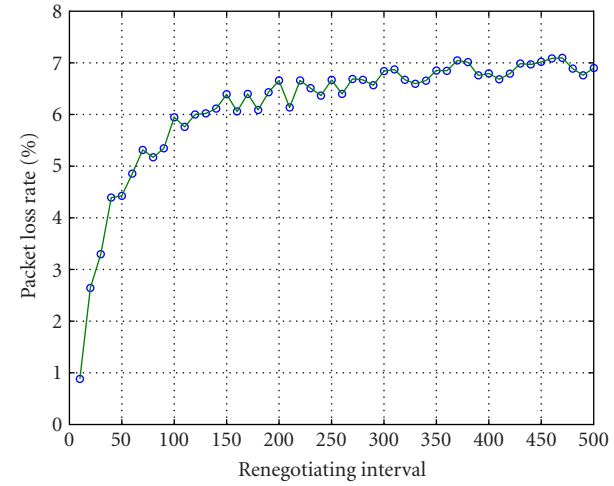
Figure 2: Performance comparison (the test trace file is Star Wars and the packet size is 100 bytes): (a) average token bucket size, (b) token drop rate, and (c) packet loss rate. The circles denote specific data at renegotiating intervals and the solid lines denote the interpolated values.



(a)



(b)



(c)

Figure 3: Performance comparison (the test trace file is Terminator 2 and the packet size is 100 bytes): (a) average token bucket size, (b) token drop rate, and (c) packet loss rate. The circles denote specific data at renegotiating intervals and the solid lines denote the interpolated values.

4. EXPERIMENTAL RESULTS

In the experiment, the test trace files are Star Wars ($240 * 352$ size) and Terminator 2 (QCIF size) encoded by MPEG-1 [22, 23, 24], whose lengths are 40 000 frames. The encoding structure is IBBPBBPBBPBB (i.e., 1GOP consists of 12 frames), and I-frames, P-frames, and B-frames are encoded with quantization parameters 10, 14, and 18, respectively. The encoding frame rate is 25 frames per second. As a result, the output traffics are VBR and their statistical properties are summarized in Table 1. The variables and threshold values of the proposed schemes are determined as follows.

(i) The tolerable maximum packet loss rate in (5) is set to 3%.
(ii) The smoothing buffer size is set to the average value of two GOPs (223516 bytes for Star Wars and 261714 bytes for Terminator 2).
(iii) The basic renegotiating interval is set to 10 GOPs.
(iv) The tested packet sizes are 100 bytes or 400 bytes.
(v) The reference window size is set to the whole frame number (40 000 frames).
(vi) The weighting factors α and β in (2) and (3) are set to 1.

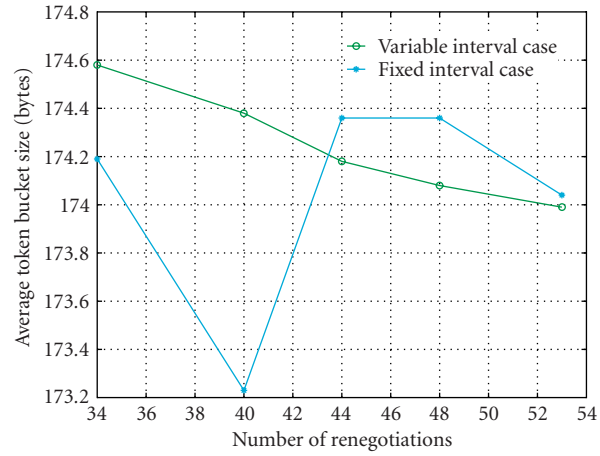
To compare the performance of the proposed QoS renegotiating schemes, we use average token drop rate, average token bucket size, and token filling rate as the network utilization measure, and packet loss rate is employed as the video quality degradation measure.

4.1. Fixed renegotiating interval case

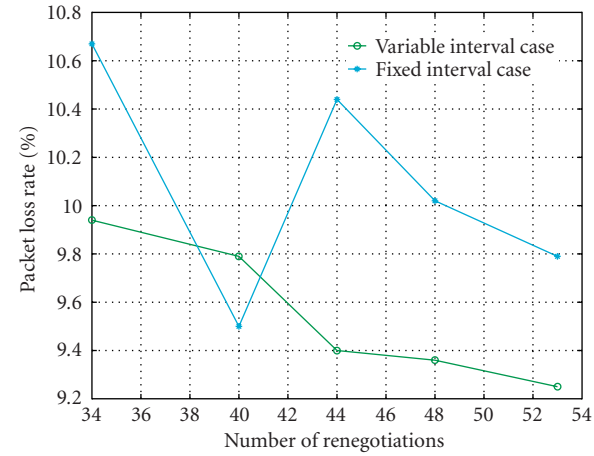
The performance comparison with respect to various fixed renegotiating intervals is shown in Tables 2, 3, 4, and 5, and Figures 2 and 3. It is observed that the average token bucket size is reduced by about 11% as the renegotiating interval decreases while the average token filling rate is almost the same for all renegotiating intervals (it can be understood since token bucket size is determined relatively by comparing the standard deviation in the reference window with that in the current renegotiating interval, see (2)). As a result, the network utilization can be improved. Furthermore, token drop rate is reduced by about 90% and packet loss rate is reduced by about 75% when the renegotiating interval is set to 10 GOPs. The same results are observed regardless of the packet size. It means that the waste of network resource caused by the dropped tokens and the video quality degradation caused by the lost packets can be significantly reduced. However, it is observed in Figures 2 and 3 that the average token bucket size and packet loss rate locally fluctuate even though the average renegotiating interval decreases. As mentioned earlier, one of the reasons is that inappropriate renegotiating instants may occur when the renegotiating interval is fixed.

4.2. Variable renegotiating interval case

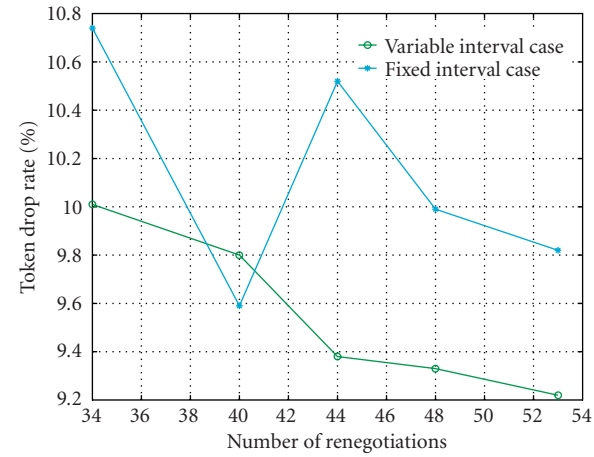
In this section, variable renegotiating time interval case is examined. The experimental results are summarized in Tables 6, 7, 8 and 9, and Figure 4. It is observed in Tables 6 and 7 that the average token bucket size is almost the same, while token



(a)



(b)



(c)

Figure 4: Performance comparison between variable renegotiating interval scheme and fixed renegotiating interval scheme (the test trace file is Star Wars and the maximum packet size is 100 bytes): (a) average token bucket size, (b) packet loss rate, and (c) token drop rate.

Table 1: Statistical properties of test MPEG trace files.

Trace files	Minimum value (bytes)	Maximum value (bytes)	Average (bytes)	Standard deviation (bytes)
Star Wars	275	124816	9313.2	12902.725
Terminator 2	312	79560	10904.75	10158.031

Table 2: Performance comparison of the fixed renegotiating interval case when the packet size is 100 bytes and the test trace file is Star Wars encoded by MPEG-1.

Fixed renegotiating interval	With renegotiations							Without renegotiation
Interval (GOPs)	10	20	50	90	130	200	300	3330
Avg. token filling rate	93.59	93.60	93.67	93.59	93.67	93.76	93.54	94
Avg. token bucket size (bytes)	166.74	169.34	172.81	173.80	176.20	177.44	177.58	185.06
Token drop rate (%)	1.78	4.92	9.00	10.08	11.89	12.78	12.70	17.26
Packet loss rate (%)	1.77	4.90	8.91	10.10	11.81	12.62	12.74	16.90

Table 3: Performance comparison of the fixed renegotiating interval case when the packet size is 400 bytes and the test trace file is Star Wars encoded by MPEG-1.

Fixed renegotiating interval	With renegotiations							Without renegotiation
Interval (GOPs)	10	20	50	90	130	200	300	3330
Avg. token filling rate	23.75	23.78	23.76	23.82	23.73	23.74	23.70	24
Avg. token bucket size (bytes)	42.35	43.02	43.90	44.21	44.76	45.22	45.08	47.02
Token drop rate (%)	1.68	4.81	8.70	9.97	11.50	12.38	12.36	17.21
Packet loss rate (%)	1.75	4.73	8.71	9.79	11.62	12.49	12.60	16.39

Table 4: Performance comparison of the fixed renegotiating interval case when the packet size is 100 bytes and the test trace file is Terminator 2 encoded by MPEG-1.

Fixed renegotiating interval	With renegotiations							Without renegotiation
Interval (GOPs)	10	20	50	90	130	200	300	3330
Avg. token filling rate	109.53	109.54	109.47	109.49	109.49	109.52	109.56	110
Avg. token bucket size (bytes)	206.17	208.81	211.29	212.50	213.47	214.30	214.66	215
Token drop rate (%)	0.91	2.69	4.42	5.36	6.04	6.71	6.90	8.37
Packet loss rate (%)	0.88	2.64	4.43	5.34	6.02	6.66	6.84	8.25

Table 5: Performance comparison of the fixed renegotiating interval case when the packet size is 400 bytes and the test trace file is Terminator 2 encoded by MPEG-1.

Fixed renegotiating interval	With renegotiations							Without renegotiation
Interval (GOPs)	10	20	50	90	130	200	300	3330
Avg. token filling rate	27.74	27.76	27.80	27.77	27.77	27.88	27.80	28
Avg. token bucket size (bytes)	52.47	53.18	53.77	54.15	54.28	54.64	54.62	55
Token drop rate (%)	0.86	2.64	4.41	5.26	5.94	6.78	6.86	8.33
Packet loss rate (%)	0.88	2.57	4.20	5.17	5.82	6.30	6.66	7.48

Table 6: Performance comparison between variable renegotiating interval case and fixed renegotiating interval case when the test trace file is Star Wars encoded by MPEG-1 and the maximum packet size is 100 bytes.

	Variable renegotiating approach				Fixed renegotiating approach			
μ	Number of renegotiation	Average token drop rate (%)	Average token bucket size (bytes)	Average packet loss rate (%)	Number of renegotiation	Average token drop rate (%)	Average token bucket size (bytes)	Average packet loss rate (%)
10	53	9.22	173.99	9.25	53	9.82	174.04	9.79
20	48	9.33	174.08	9.36	48	9.99	174.36	10.02
30	44	9.38	174.18	9.40	44	10.52	174.26	10.44
40	40	9.80	174.38	9.79	40	9.59	173.23	9.50
50	34	10.01	174.58	9.94	34	10.74	174.19	10.67

Table 7: Renegotiating time instants of variable renegotiating interval case and fixed renegotiating interval case when the test trace file is Star Wars encoded by MPEG-1 and the maximum packet size is 100 bytes.

Method	QoS renegotiating instants (frame number)
Variable interval	0, 600, 840, 2280, 2880, 3000, 3840, 3960, 4680, 5400, 5760, 7200, 7320, 7920, 8280, 9120, 10080, 10560, 11520, 15120, 15840, 17880, 19440, 20160, 20760, 21240, 21720, 21840, 22320, 22680, 23760, 24840, 24960, 25800, 26400, 27240, 28920, 29520, 29640, 29760, 30120, 30720, 31320, 33360, 33600, 33840, 35400, 35520, 36480, 37560, 37920, 38280, 38640
Fixed interval	0, 732, 1464, 2196, 2928, 3660, 4392, 5124, 5856, 6588, 7320, 8052, 8784, 9516, 10248, 10980, 11712, 12444, 13176, 13908, 14640, 15372, 16104, 16836, 17568, 18300, 19032, 19764, 20496, 21228, 21960, 22692, 23424, 24156, 24888, 25620, 26352, 27084, 27816, 28548, 29280, 30012, 30744, 31476, 32208, 32940, 33672, 34404, 35136, 35868, 36600, 37332, 38064

Table 8: Performance comparison between variable renegotiating interval case and fixed renegotiating interval case when the test trace file is Terminator 2 encoded by MPEG-1 and the maximum packet size is 100 bytes.

	Variable renegotiating approach				Fixed renegotiating approach			
μ	Number of renegotiation	Average token drop rate (%)	Average token bucket size (bytes)	Average packet loss rate (%)	Number of renegotiation	Average token drop rate (%)	Average token bucket size (bytes)	Average packet loss rate (%)
10	46	5.16	212.25	5.13	46	5.19	212.08	5.14
20	44	5.54	212.73	5.47	44	5.70	212.47	5.64
30	43	5.60	213.83	5.51	43	5.76	212.66	5.73
40	43	5.60	212.79	5.51	43	5.76	212.6	5.73
50	43	5.60	212.79	5.51	43	5.76	212.6	5.73

Table 9: Renegotiating time instants of variable renegotiating interval case and fixed renegotiating interval case when the test trace file is Terminator 2 encoded by MPEG-1 and the maximum packet size is 100 bytes.

Method	QoS renegotiating instants (frame number)
Variable interval	0, 120, 480, 1080, 1800, 2400, 3720, 5040, 5520, 5880, 7920, 8160, 8880, 9960, 10680, 12000, 12480, 13440, 14760, 15240, 15960, 16680, 17880, 18720, 19560, 20400, 20880, 22080, 23280, 24120, 24600, 25560, 26760, 27000, 27600, 28920, 29040, 32160, 32760, 33120, 33840, 34800, 35160, 35760, 36840, 38040
Fixed interval	0, 852, 1704, 2556, 3408, 4260, 5112, 5964, 6816, 7668, 8520, 9372, 10224, 11076, 11928, 12780, 13632, 14484, 15336, 16188, 17040, 17892, 18744, 19596, 20448, 21300, 22152, 23004, 23856, 24708, 25560, 26412, 27264, 28116, 28968, 29820, 30672, 31524, 32376, 33228, 34080, 34932, 35784, 36636, 37488, 38340

Table 10: Performance comparison between the proposed algorithm and bandwidth renegotiating scheme (test trace file is Star wars).

Number of renegotiations	Proposed algorithm		Channel bandwidth renegotiating algorithm	
	Token drop rate (%)	Packet loss rate (%)	Token drop rate (%)	Packet loss rate (%)
53	9.22	9.25	9.79	9.89
48	9.33	9.36	9.89	10.04
44	9.38	9.40	9.94	10.22
40	9.80	9.79	10.56	10.42
34	10.0	9.94	10.62	10.62

Table 11: Performance comparison between the proposed algorithm and bandwidth renegotiating scheme (test trace file is Terminator 2).

Number of renegotiations	Proposed algorithm		Channel bandwidth renegotiating algorithm	
	Token drop rate (%)	Packet loss rate (%)	Token drop rate (%)	Packet loss rate (%)
46	5.16	5.13	5.59	5.46
44	5.54	5.47	5.99	5.83
43	5.60	5.51	6.04	5.88
43	5.60	5.51	6.04	5.88
43	5.60	5.51	6.04	5.88

drop rate and packet loss rate are reduced by 8.6% and 7.5%, respectively, when the number of renegotiations is changed from 43 to 46. Thus, the waste of network resource can be reduced and the video quality degradation caused by the lost packets can be decreased too. In addition, it is observed that average token drop rate, average token bucket size, and token filling rate monotonically decrease while those of fixed renegotiating approach locally fluctuate. We can see the obvious differences of the renegotiating time instants in Tables 7 and 8. It means that we can predict the traffic characteristics more accurately by the interpolation method when μ changes. Hence, we can conclude that variable renegotiating approach can determine the renegotiating instants more effectively than fixed renegotiating approach at the cost of the increased computational complexity.

4.3. Performance comparison with bandwidth renegotiating schemes

In this section, we compare the proposed algorithm with bandwidth renegotiating algorithms. Actually, it is not easy to simply compare the performance with bandwidth renegotiating algorithms since they provide the deterministic services and consider the different network situations. Thus, we implemented the channel bandwidth renegotiating scheme by token bucket model with a piecewise constant token filling rate and a fixed token bucket size (it is set to the average value of the proposed algorithm) and then tested various renegotiating interval cases. The experimental results are summarized in Tables 10 and 11, and Figure 5. As shown in the tables and figure, we observe that the proposed algorithm can reduce both the packet loss rate and the token drop rate. The reason is that the proposed algorithm treats token bucket size as well as token filling rate as control variables while the bandwidth renegotiating schemes consider only token filling rate as a control variable.

We would like to give some remarks on the experimental results. We obtain Figure 6 when the histograms of video traffics are drawn. They look like Poisson distributed although we assume Gaussian distribution for simplicity. This mismatch can cause some errors, and the basic renegotiating interval may also be related to the errors. As the length of basic renegotiating interval becomes small, the performance may be improved at the expense of higher computational complexity.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented effective token bucket parameter renegotiating schemes for streaming video over network supporting QoS renegotiations. Two approaches, fixed renegotiating interval case and variable renegotiating interval case, are examined. The experimental results showed that the average token bucket size and the packet loss rate are significantly reduced as the number of renegotiations increases. Furthermore, variable renegotiating interval case avoids the inappropriate renegotiating instants of fixed renegotiating interval case at the cost of the increased computational complexity. Based on these observations, we can conclude that the proposed flexible QoS renegotiating approach can improve the network utilization compared to the bandwidth renegotiating approach and is a promising technique for the effective streaming video. On the other hand, if Tables 6 and 8 are stored as metadata in database, we can estimate the average token bucket model parameters of the new video on-demand request by linear interpolation method with a low computational complexity. Basically, the information may be very

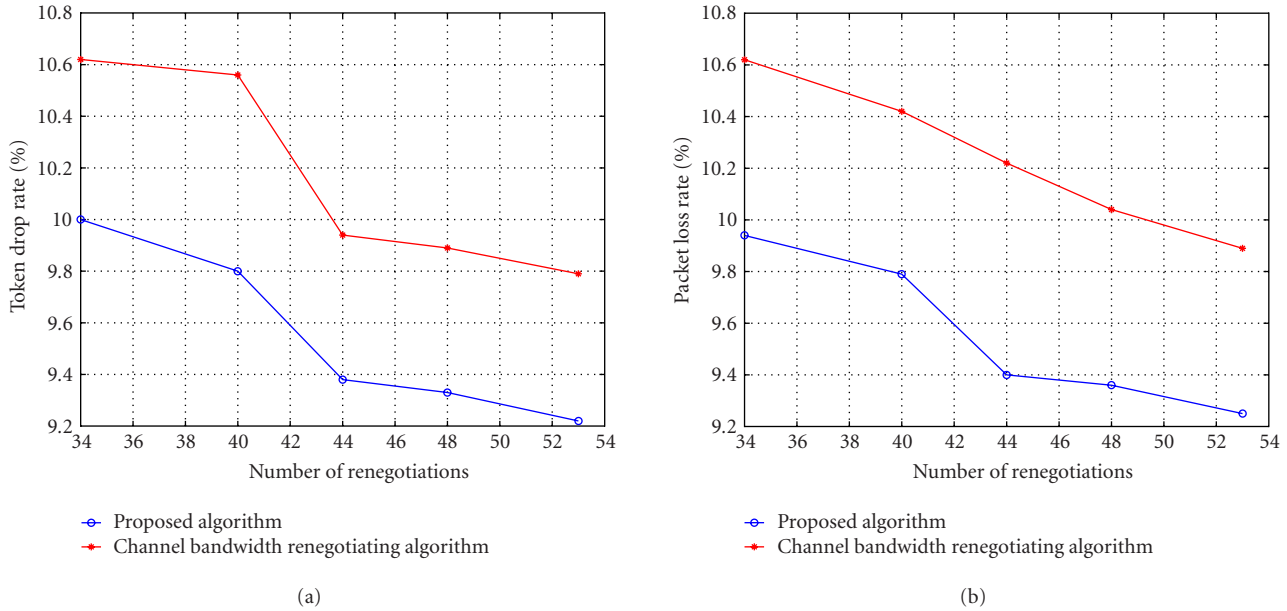


Figure 5: Performance comparison between the proposed algorithm and bandwidth renegotiating scheme (when the test trace file is Star Wars and packet size is 100 bytes): (a) token drop rate and (b) packet loss rate.

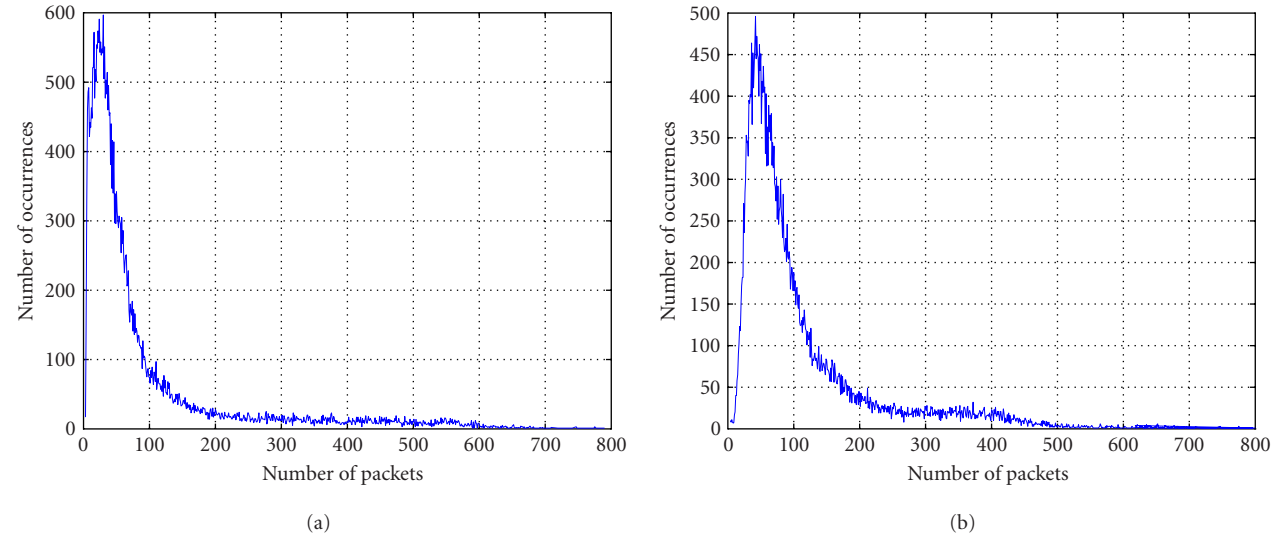


Figure 6: Histogram of test video traffics: (a) Star Wars and (b) Terminator 2.

helpful to design a simple but quite effective call admission control algorithm. For the complete solution, we need the rate shaping/adaptation algorithm to adjust the compressed video bitstream when the QoS requests are sometimes rejected which is under our current investigation.

ACKNOWLEDGMENT

This work is supported by the University Fundamental Research Program supported by the Ministry of Information & Communication of the Republic of Korea.

REFERENCES

[1] ISO/IEC 13818 (MPEG-2), "Generic coding of moving pictures and associated audio information," November 1994.
[2] ISO/IEC JTC 1/SC 29/WG 11 N4030, "Overview of the MPEG-4 standard," March 2001.
[3] ITU-T Recommendation H.261, "Video Codec for Audio Visual Services at $p * 64$ kbits/s," March 1993.
[4] ITU-T Recommendation H.263 version 2, "Video coding for low bitrate communication," January 1998.
[5] T. V. Lakshman, A. Ortega, and A. R. Reibman, "VBR video: Tradeoffs and potential," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 952–973, 1998.

[6] Z.-L. Zhang, J. Kurose, J. D. Salehi, and D. Towsley, "Smoothing, statistical multiplexing, and call admission control for stored video," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 1148–1166, 1997.
[7] M. Grossglauser, S. Keshav, and D. C. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 741–755, 1997.
[8] A. Mohammad, "Using adaptive linear prediction to support real-time VBR video under RCBR network service model," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 635–644, 1998.
[9] T.-Y. Kim, B.-H. Roh, and J.-K. Kim, "Bandwidth renegotiation with traffic smoothing and joint rate control for VBR MPEG video over ATM," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 10, no. 5, pp. 693–703, 2000.
[10] H. Song and K. M. Lee, "Adaptive rate control algorithms for low-bit-rate video under the networks supporting bandwidth renegotiation," *Signal Processing: Image Communication*, vol. 17, no. 10, pp. 759–779, 2002.
[11] H. Zhang and E. W. Knightly, "RED-VBR: A renegotiation-based approach to support delay-sensitive VBR video," *ACM Multimedia Systems Journal*, vol. 5, no. 3, pp. 164–176, 1997.
[12] J. D. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 397–410, 1998.
[13] M. Wu, R. A. Joyce, H.-S. Wong, L. Guan, and S.-Y. Kung, "Dynamic resource allocation via video content and short-term traffic statistics," *IEEE Trans. Multimedia*, vol. 3, no. 2, pp. 186–199, 2001.
[14] B. V. Patel and C. C. Bisdikian, "End-station performance under leaky bucket traffic shaping," *IEEE Network*, vol. 10, no. 5, pp. 40–47, 1996.
[15] D. Grossman, "Definition of VBR service," Contribution ATM Forum/94-0816, September 1994.
[16] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," IETF RFC 2212, September 1997.
[17] S. Verma, R. K. Pankaj, and A. Leon-Garcia, "Call admission and resource reservation for guaranteed QoS services in Internet," *Computer Communication*, vol. 21, no. 4, pp. 362–374, 1998.
[18] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," IETF RFC 2475, December 1998.
[19] J. Glasmann, M. Czermin, and A. Riedl, "Estimation of token bucket parameters for videoconferencing systems in cooperate networks," in *International Conference on Software, Telecommunications and Computer Networks*, Trieste, October 2000.
[20] N. Farber, K. Stuhlmuller, and B. Girod, "Analysis of error propagation in hybrid video coding with application to error resilience," in *Proceedings of IEEE International Conference on Image Processing*, Kobe, Japan, October 1999.
[21] J.-G. Kim, J. Kim, J. Shin, and C.-C. J. Kuo, "Coordinated packet level protection employing corruption model for robust video transmission," in *SPIE Proc. of Visual Communication and Image Processing*, San Jose, Calif, USA, January 2001.
[22] Berkeley Multimedia Research Center, ftp://mm-ftp.cs.berkeley.edu/pub/.
[23] Bellcore, ftp://ftp.telecordia.com/pub/vbr/video/trace/.
[24] KAIST, http://viscom.kaist.ac.kr/.

Hwangjun Song received his B.S. and M.S. degrees from the Department of Control and Instrumentation, School of Electrical Engineering, Seoul National University, Korea, in 1990 and 1992, respectively, and his Ph.D. degree in electrical engineering systems, University of Southern California, Los Angeles, Calif., USA, in 1999. He was a Research Engineer at LG Industrial Lab., Korea, in 1992. From 1995 to 1999, he was a Research Assistant in SIPI (Signal and Image Processing Institute) and IMSC (Integrated Media Systems Center), University of Southern California. Since 2000, he has been a faculty member with the School of Electronic and Electrical Engineering, Hongik University, Seoul, Korea. His research interests include multimedia signal processing and communication, image/video compression, digital signal processing, network protocols necessary to implement functional image/video applications, control system and fuzzy-neural system.

Dai-Boong Lee received his B.S. degree from Hongik University, Seoul, Korea, in 2002, where he is currently working toward his M.S. degree in Multimedia Communication System Lab., School of Radio Science and Communication Engineering. His research interests include packet scheduling, quality-of-service network, Int/Diffserv, network resource renegotiation algorithm, network management, and visual information processing.

EURASIP Journal on Applied Signal Processing 2004:2, 290–303

Error Resilient Video Compression Using Behavior Models

Jacco R. Taal
Information and Communication Theory Group, Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
Email: j.r.taal@ewi.tudelft.nl

Zhibo Chen
Imaging Technology Group, IMNC, Sony Corporation, 6-7-35 Kitashinagawa, Shinagawa-Ku, Tokyo 141-0001, Japan
Email: chenzhibo@tsinghua.org.cn

Yun He
Video Communication Research Group, Electronic Engineering Department, Tsinghua University, 11-425 East Main Building, 100084 Beijing, China
Email: hey@video.mdc.tsinghua.edu.cn

R. (Inald) L. Lagendijk
Information and Communication Theory Group, Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
Email: r.l.lagendijk@ewi.tudelft.nl

Received 1 December 2002; Revised 26 September 2003

Wireless and Internet video applications are inherently subjected to bit errors and packet errors, respectively. This is especially so if constraints on the end-to-end compression and transmission latencies are imposed. Therefore, it is necessary to develop methods to optimize the video compression parameters and the rate allocation of these applications that take into account residual channel bit errors. In this paper, we study the behavior of a predictive (interframe) video encoder and model the encoders behavior using only the statistics of the original input data and of the underlying channel prone to bit errors. The resulting data-driven behavior models are then used to carry out group-of-pictures partitioning and to control the rate of the video encoder in such a way that the overall quality of the decoded video with compression and channel errors is optimized.

Keywords and phrases: behavior model, rate distortion, video coding, error resilience.

1. INTRODUCTION

Although the current video compression techniques can be considered mature, there are still many challenges in the design and operational control of compression techniques for end-to-end quality optimization. This is in particular true in the context of unreliable transmission media such as the Internet and wireless links. Conventional compression techniques such as JPEG and MPEG were designed with error-free transmission of the compressed bitstream in mind. With such unreliable media, not all bit or packet errors may be corrected by retransmissions or forward error correction (FEC). Depending on the kind of channel coder, residual channel errors may be present in the bitstream after channel decoding. In most practical packet network systems, packet retransmission corrects for some, but not all, packet losses. Classic rate control, such as TM.5 in MPEG [1], can be used to control the video encoder according to the available bit rate offered by the channel coder; adaptation to the bit error rate by inserting intracoded blocks is nevertheless not incorporated in TM.5. Other methods that control the insertion of intracoded blocks exist [2].

Three classes of error resilient source coding techniques that deal with error prone transmission channels may be distinguished. The first well-known approach is joint source-channel coding, which aims to intimate integration of the source and channel coding algorithms [3, 4]. Although this intimate integration brings several advantages to the end-to-

end quality optimization, it comes at the price of a significant complexity increase. Furthermore, nearly all of these approaches only work with specific or nonstandard network protocols and with a specific video encoder and/or decoder.

The second class represents many approaches where the source coder has no (or limited) control of the network layer. It is important to understand that these approaches can not be generally optimal since the channel coder and the source coder are not jointly optimized. Since there is no joint optimization, the only thing the source coder can do is to adapt its own settings according to the current behavior of the network layer. In many applications, joint optimization is impossible because none of the standard network protocols (IP, TCP, and UDP) support this. Even though the source coder has no or limited control over the network layer, the rate control algorithm can adapt to the available bit rate and to the amount of residual bit errors or packet losses. Such a control algorithm needs a model describing the effects of bit errors or packet losses on the overall distortion.

The third class contains the approaches advocated in [5, 6]. In these approaches, the best properties of the first two classes are combined. Here, the authors propose to limit the integration to joint parameter optimization, so that there is no algorithmic integration. In previous work at Delft University of Technology [7], an efficient overall framework was proposed for such joint parameter optimization from a quality-of-Service (QoS) perspective. This framework requires high-level and abstract models describing the behavior of source and channel coding modules. However, this framework had not yet been tested with a real video coder and with a real behavior model.

In this paper, we propose such a behavior model for describing source-coding characteristics, giving some information about the channel coder. Although this model is designed to be used in a QoS setup, it may also be used to optimize the encoders settings when we only have knowledge of, but no control over, the current channel (as a second class approach).

With this behavior model, we can predict the behavior of a source coder in terms of the image quality related to the channel coder parameters: the bit rate, the bit error rate (BER), and the latency. To be applicable in a real-time and perhaps low power setup, the model itself should have a low complexity and should not require that many frames have to reside in a buffer (low latency).

We evaluate the behavior models with one type of progressive video coder. However, we believe that other coders can be described fairly easily with our methods as well, since we try to describe the encoders at the level of behavior rather than at a detailed algorithmic or implementation level. In Section 2, we first discuss our combined source-channel coding system; the problem we wish to solve, and we describe the source and channel coders on a fairly high abstraction level. From these models, we can formulate the end-to-end quality control as an optimization problem, which we will discuss in Section 3. Section 4 describes in depth the construction of the proposed models. In Section 5, our models are validated in a simulation where a whole group of pictures (GOP) were transmitted over an error prone channel. Section 6 concludes this paper with a discussion.

2. PROBLEM FORMULATION

To optimize the end-to-end quality of compressed video transmission, one needs to understand the individual components of the link. This understanding involves knowledge of the rate distortion performance and the error resilience of the video codec, of the error correcting capabilities of the channel codec, and possibly of parameters such as delay, jitter, and power consumption. One of the main challenges in attaining an optimized overall end-to-end quality is the determination of the influence of the individual parameters controlling the various components. Especially because the performances of various components depend on each other, and the control of these parameters is not straightforward.

In [4, 5, 8, 9], extensive analyses of the interaction and trade-offs between source and channel coding parameters can be found. A trend in these approaches is that the underlying components are modeled at a fairly high abstraction level. The models are certainly independent of the actual hardware or software implementation but they also become more and more independent of the actual compression or source coding algorithm used. This is in strong contrast to the abundance of joint source channel coding approaches, which typically optimize a particular combination of source and channel coders, utilizing specific internal algorithmic structures and parameter dependencies. Although these approaches have the potential to lead to the best performance, their advantages are inherently limited to the particular combination of coders and to the conditions (source and channel) under which the optimization was carried out.

In this paper, we refrain from the full integration of source and channel codecs (i.e., the joint source-channel coding approach) but we keep the source and channel coders as much separate as possible.

The interaction between source and channel coders and, in particular, the communication of key parameters is encapsulated in a QoS framework. The objective of the QoS framework is to structure the communication context parameters between OSI layers. In the scope of this paper, the context can be defined not only by radio/Internet channel conditions, but also by the demands of the application or device concerning the quality or the complexity of the video encoding. Here we discuss only the main outline of the QoS interface. A more detailed description of the interface can be found in the literature (see [6, 7]).

Figure 1 illustrates the QoS Interface concept [7]. The source and channel coders operate independent of each other, but are both under the control of QoS controllers. The source coder encodes the video data, thereby reducing the needed bit rate. The channel coder protects this data. It decreases the BER, thereby effectively reducing the bit rate available for source coding and increasing the latency. The QoS controller of the source coder communicates the key parameters—in this case, the bit rate, the BER, and latency—

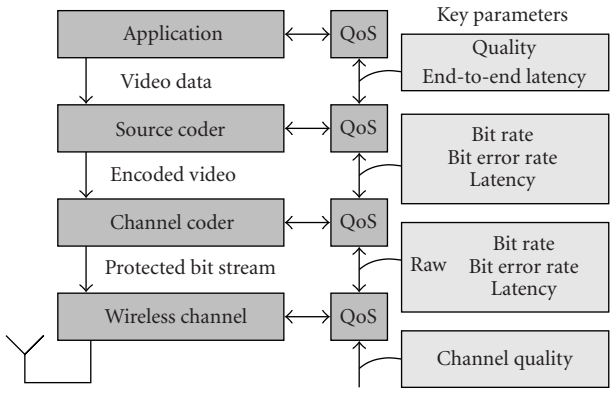


Figure 1: QoS concept: different (OSI)layers are not only communicating their payloads, they are also controlled by QoS controllers that mutually negotiate to optimize the overall performance.

with the QoS controller of the channel coder. Based on the behavior description of the source and channel coding modules, the values of these parameters are optimized by the QoS controller. In a practical system, this optimization takes into account context information about the application (e.g., maximum latency) and about the channel (e.g., throughput at the physical layer). The application may set constraints on the operation of the lower layers, for instance, on the power consumption or the delay. In this paper, we assume that the only constraint set by the application is the end-to-end delay T_a.

In order to implement the QoS Interface/controller concept, the following three problems need to be solved.

(i) The key parameters must be optimized over different (OSI) layers. We have developed the "adaptive resource contracts" (ARC) approach for solving this problem. ARC exchanges the key parameters between two layers such that after a negotiation phase, both layers agree on the values of these parameters. These key parameters represent the trade-offs that both layers have made to come to a joint solution of the optimization. A detailed discussion of ARC falls outside the scope of this paper. We refer to [6, 7, 10].

(ii) The behavior of the source and channel coders should be modeled parametrically such that joint optimization of the key parameters can take place. At the same time, an internal controller that optimizes the performance of the source and channel coders independently, given the already jointly optimized key parameters, should be available. The emphasis in this paper is on the modeling of the video coder behavior.

(iii) An optimization procedure should be designed for selecting the parameters internal to the video codec, given the behavior model and the key parameters. We do not emphasize this aspect of the QoS interface in this paper as we believe that the required optimization procedure can be based on related work as that in [11].

In previous work and analyses [6, 7], the source coder was modeled as a *progressive encoder*, which means that with every additionally transmitted bit, the quality of the received decoded information increases. Therefore, the most important information is encoded at the beginning of the data stream, and the least important information is encoded at the end. In principle, we believe that any progressive encoder can be described with our models. To keep things simple from a compression point of view, we use the common interframe coding structure (with one interframe per GOP, multiple predictively encoded interframes, and no bidirectional encoded frames). The actual encoding of the (difference) frames is done by a JPEG2000 (see [12]) encoder, which suits our demand for progressive behavior. Figure 2 shows the typical block diagram of this JPEG2000-based interframe coder. In this paper, we exclude motion compensation of interframes for simplicity reasons. The internal parameters for this video encoder are the number of frames in a GOP N, and the bit rates r_i for the individual frames. Symbols X_i and X_{i-1} denote the current frame and the previous frame, $\overline{X}$ denotes a decoded frame, and $\tilde{X}$ denotes a decoded frame at the decoder side, possibly with distortions caused by residual channel errors. Symbols $\tilde{D}_{\mathrm{q}}$ and $\tilde{D}_{\mathrm{e}}$ denote the quantization distortion and the distortions caused by residual channel errors (named "channel-induced distortion" hereafter), respectively.

In this work, the channel coder is defined as an abstract functional module with three interface parameters. The channel coder has knowledge of the current state of the channel which it is operating on. Therefore, it can optimize its own internal settings using behavior models. Such a channel coder may use different techniques like FEC and automatic repeat requests (ARQ) to protect the data at the expense of added bit rate and increased delay (latency). The exact implementation is nevertheless irrelevant for this paper. From here we will assume that the error protection is not perfect because of latency constraints; therefore the residual BER may be non zero. The behavior models can be obtained by straightforward analysis of the channel coding process [5].

3. SOURCE ENCODER OPTIMIZATION CRITERION

At this point, we assume that we have a behavior model for our video encoder. The development of this behavior model is the subject of Section 4. Given the behavior model, we can minimize the average end-to-end distortion $\hat{D}$ given the constraints imposed by the QoS interface. In our work, the QoS Interface negotiates three key parameters between source and channel coder, namely, $\{R, \mathrm{BER}, T_c\}$, with

(i) R: the available bit rate for source coding (average number of bits per pixel);

(ii) the residual BER: the average bit error rate after channel decoding;

(iii) T_c: the average time between handing a bit to the channel encoder, and receiving the same bit from the channel decoder.

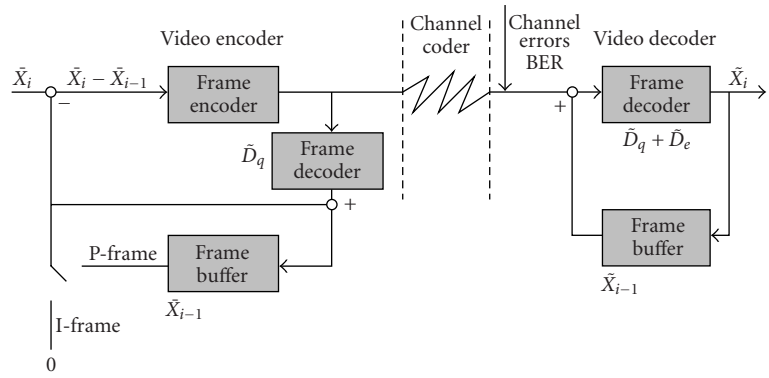


Figure 2: Simple video coding scheme. The "frame encoder" and "frame decoder" blocks represent the single frame encoder and decoder. The "frame buffer" is needed for the predictively encoded interframes.

The resulting source coding optimization problem now becomes the minimization of the distortion D, which can be formulated as follows:

$$\min_{I_{\text{src}}} D(I_{\text{src}} \mid \{R, \text{BER}, T_c\}). \tag{1}$$

Here, I_{src} denotes the set of internal source coder parameters over which the performance of the encoder must be optimized, given the key parameters $\{R, \text{BER}, T_c\}$. The actual set of internal parameters to be considered depends on the encoder under consideration and the parameters included in the encoders behavior model. In this paper, we consider the optimization of the following internal parameters:

(i) N, the length of the current GOP. Each GOP starts with an intraframe and is followed by $N-1$ predictively encoded interframes;

(ii) $\vec{r} = \{r_0, r_1, \ldots, r_{N-1}\}$: the target bit rate for each individual frame in a GOP.

The encoder parameter N relates to the coding efficiency and the robustness of the compressed bitstream against the remaining errors. The larger is N, the higher the coding efficiency, because more interframes are encoded. At the same time, the robustness of the stream is lower due to the propagation of decoded transmission errors.

On the other hand, in order to optimize the settings $\{N, \vec{r}\}$ for $N_{\max}$ frames, these $N_{\max}$ frames have to be buffered, thereby introducing a latency. In our approach, the QoS interface prescribes the maximum end-to-end latency T_a (seconds), and we assume that the channel coder will have an end-to-end latency of T_c (seconds), from the channel encoder to the channel decoder, including transmission. Analysis of the whole transmission chain gives the following expression for the total end-to-end latency:

$$T_a = \frac{N-1}{f_r} + T_e + T_c + \frac{B}{R}, \tag{2}$$

where f_r is the frame rate of the video sequence that is encoded and T_e is the upper bound of the time it takes to encode a frame. Finally B/R is the transmission time for one frame B/R; the maximal number of bits to describe a frame divided by the channel coding bit rate R.

We can now find an expression for the maximal number of frames that can be in the buffer while still meeting the end-to-end latency constraints T_a. Clearly B/R is only known after allocating the rate for each frame. We suggest taking the worst case value for B (i.e., calculated from the maximal bit rate setting). The same goes for T_E where we suggest to take the worst case encoding time per frame,

$$N_{\max} = 1 + \left(T_a - T_e - T_c - \frac{B}{R}\right) f_r. \tag{3}$$

In each frame i, two kinds of distortion are introduced: (1) the quantization error distortion, denoted by D_q and (2) the channel-induced distortion caused by bit errors in the received bitstream, denoted by D_e. With our optimization problem, we aim to minimize the average distortion, which is the sum of individual distortions of a GOP divided by the length of the group:

$$D_{\text{GOP}} = \frac{1}{N} \sum_{i=0}^{N-1} \{D_{\text{q}}(r_i) + D_{\text{e}}(r_i, \text{BER})\}. \tag{4}$$

Following [5], we assume that D_q and D_e within one frame are mutually independent. Although (4) is a simple additive distortion model, the distortion of a frame is still dependent on that of the previous frames because of the interframe prediction. Therefore, in our models, we have to take into account the propagation of quantization and channel-induced distortions.

Taking the above parameters into account, we can now rewrite (1) as the following bit rate allocation problem:

$$\begin{aligned} \vec{r}_{\text{opt}}, N_{\text{opt}} &\longleftarrow \min_{\vec{r},N} D_{\text{GOP}}(\vec{r}, N \mid \text{BER}) \\ &= \min_{N} \left\{ \min_{\vec{r}} \frac{1}{N} \sum_{i=0}^{N-1} D_{\text{q}}(r_i) + D_{\text{e}}(r_i, \text{BER}) \right\} \end{aligned} \tag{5}$$

subject to

$$\frac{1}{N}\sum_{i=1}^{N-1} r_i = R, \quad N \le N_{\max}. \tag{6}$$

The approach that we follow in this paper is to optimize the bit rate allocation problem (5) and (6) based on two frame-level parametric behavior models. The first (rate distortion) model parametrically describes the relation between the variance of the quantization distortion and the allocated bit rate based on the variance of the input frames. The second (channel-induced distortion) model parametrically describes the relation between the variance of the degradations due to transmission and the decoding errors based on the variance of the input frames and the *effective* (BER).[1]

4. RATE DISTORTION MODEL

In this section, we first propose a behavior model for the rate distortion characteristics D_q of video encoders and then propose a model for distortion caused by residual channel errors including the error propagation D_e.

There are two approaches for modeling the rate distortion (RD) behavior of sources. The first approach is the analytical approach, where mathematical relations are derived for the RD functions assuming certain (stochastic) properties of the source signal and the coding system. Since these assumptions do not often hold in practice, the mismatch between the predicted rate distortion and the actual rate distortion is (heuristically) compensated for by empirical estimation. The second is the empirical approach where the RD functions are modeled through regression analysis of the empirically obtained RD data. The rate distortion model proposed in [5] is an example of an empirical model of the distortion of an entire encoder for a given bit rate.

In our work, we anticipate the real-time usage of the constructed abstract behavior models. At the same time, we want to keep the complexity of the models low. This limits the amount of *preprocessing* or *analysis* that we may do on the frames to be encoded. Therefore, we will base our behavior models on variance information only. In particular, we will use

(i) the variance of the frame under consideration denoted by $\mathrm{VAR}[X_i]$,

(ii) the variance of the difference of two consecutive frames denoted by $\mathrm{VAR}[X_i - X_{i-1}]$.

4.1. Rate distortion behavior model of intraframes

It is well known that for memoryless Gaussian distributed sources X with variance $\mathrm{VAR}[X]$, the RD function is given by

$$r(D_q) = \frac{1}{2}\log_2\left(\frac{\mathrm{VAR}[X]}{D_q}\right), \tag{7}$$

or when we invert this function by

$$D_q(r) = \mathrm{VAR}[X]2^{-2r}. \tag{8}$$

Empirical observations show that for the most common audio and video signals under small distortions, the power function $-2r$ gives an accurate model for the behavior of a compression system especially in terms of the quality gain per additional bit (in bit rate terms) spent. For instance, the power function $-2r$ leads to the well-known result that, at a sufficiently high bit rate, for most video compression systems we gain approximately 6 dB per additional bit per sample.

However, for more complicated compression systems and especially for larger distortions, the simple power function does not give us enough flexibility to describe the empirically observed RD curves, which usually give more gain for the same increase in bit rate. Since there is basically no theory to rely on for these cases without extremely detailed modeling of the compression algorithm, we instead propose to generalize (8) as follows:

$$D_q(r) = \mathrm{VAR}[X]2^{f(r)}. \tag{9}$$

The function $f(r)$ gives us more freedom to model the (RD) behavior at the price of regression analysis or online parameter estimation on the basis of observed rate distortion realizations. The choice of the kind of the function used to model $f(r)$ is a pragmatic one. We have chosen a third-order polynomial function. A first- or second-order function was simply too imprecise, while a fourth-order model did not give a significant improvement and higher-order models would defeat our objective of finding simple and generic models. Clearly there is a trade-off between precision (high order) and generality (low order).

In Figure 3, we show the (RD) curve of the experimentally obtained $\tilde{D}_q(r)$ for the JPEG2000 compression of the first frame of the Carphone sequence for bit rates between 0.05 and 1.1 bits per pixel (bpp). The solid line represents a third-order polynomial fit of $f(r)$ on the measured values. This fit is much better than the linear function $f(r) = -2r$. The following function was obtained for the first frame of the Carphone sequence:

$$D_q(r) = \mathrm{VAR}[X]2^{-4.46r^3+11.5r^2-12.7r-1.83}. \tag{10}$$

It is interesting to see how the RD curve changes for different frames of the same scene or different scenes. Figure 4 shows the RD curve for frame 1 and frame 60 of Carphone, and frame 1 of Foreman. Observe that the Carphone frames have very similar curves. The Foreman curve is shifted, but is still similar to the other two. These observations strengthen our belief that the model is generally applicable for this type of coder. Of course the $f(r)$ needs to be fitted for a particular sequence, on the other hand, we believe that a default curve $f_0(r)$ can be used to bootstrap the estimation of model parameters for other video sequences. The function $f(r)$ can then be adapted with a new RD data as the encoding continues.

[1] By "effective bit error rate" we mean the residual bit error rate, that is, the bit errors that are still present in the bitstream after channel decoding.

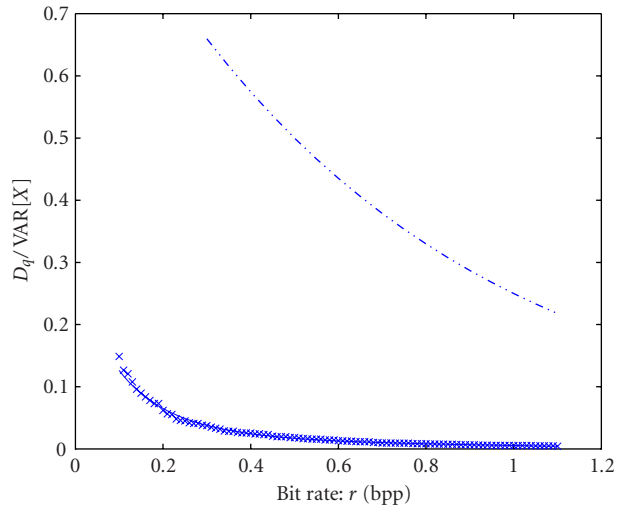


FIGURE 3: RD curve for the first frame in Carphone. The crosses (×) are the measured normalized distortions $\tilde{D}_q$ and the solid line corresponds to the fitted function $2^{f(r)}$. The dashed-dotted line corresponds to the RD model 2^{-2r}.

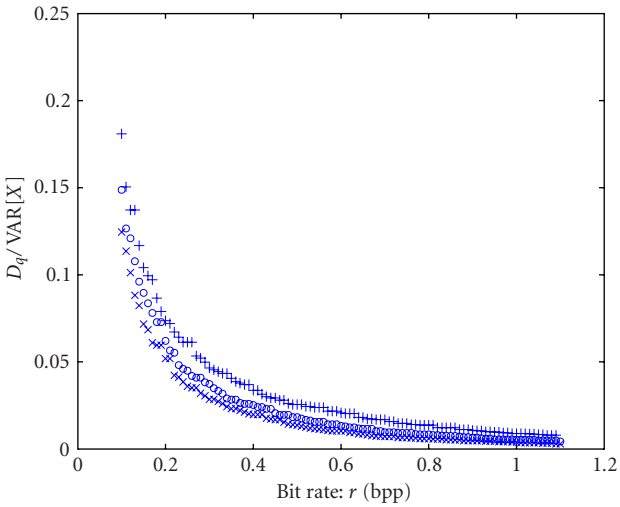


FIGURE 4: Intraframe RD curve for the first frame of Carphone (×), frame 60 of Carphone (∘), and the first frame of Foreman (+).

4.2. *Rate distortion behavior model of interframes*

For modeling the (RD) behavior of interframes, we propose to use a model similar to the one in (9), but with a different polynomial $g(r)$,

$$D_{\mathrm{q}}(r_i) = \mathrm{VAR}\left[X_i - \overline{X}_{i-1}\right]2^{g(r_i)}. \tag{11}$$

Here, $\overline{X}_{i-1}$ denotes the previously decoded frame $i-1$, whereas with intraframes, a third-order polynomial was needed to predict $f(r)$ accurately enough. With interframes, a second-order polynomial was sufficient to predict $g(r)$. The reason for this can be found in the fact that interframes are less correlated than intraframes. Therefore, $g(r)$ is more similar to the theoretical $-2r$ than $f(r)$.

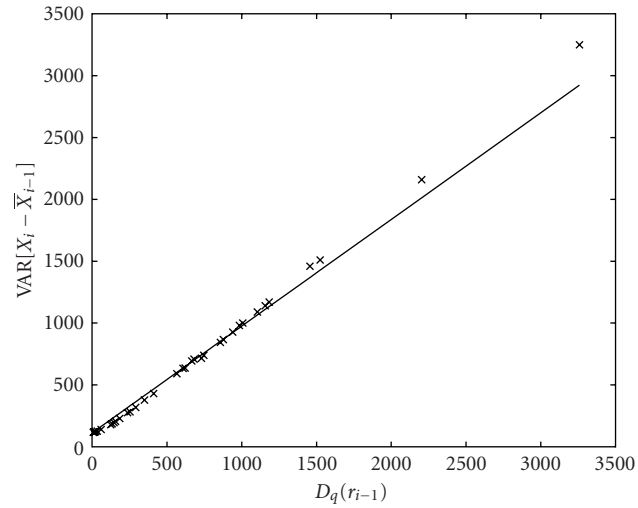


FIGURE 5: The relationship between the variance of frame difference $\mathrm{VAR}[X_i - \overline{X}_{i-1}]$ and the quantization distortion $\tilde{D}_{\mathrm{q}}(r_{i-1})$. The fitted line describes $\mathrm{VAR}[X_i - \overline{X}_{i-1}] = \mathrm{VAR}[X_i - X_{i-1}] + \kappa D_{\mathrm{q}}(r_{i-1})$.

In (11), $\mathrm{VAR}[X_i - \overline{X}_{i-1}]$ is the variance of the difference between the current frame i and the previously *encoded* frame $i-1$. Since the latter is only available *after* encoding (and thus after solving (5) and (6)), we need to approximate $\mathrm{VAR}[X_i - \overline{X}_{i-1}]$. Obviously we have

$$\begin{aligned}\mathrm{VAR}\left[X_i - \overline{X}_{i-1}\right] &= E\left[\left(X_i - \overline{X}_{i-1}\right)^2\right] \\ &= E\left[\left((X_i - X_{i-1}) - (\overline{X}_{i-1} - X_{i-1})\right)^2\right] \\ &= \mathrm{VAR}\left[X_i - X_{i-1}\right] + D_q(r_{i-1}) \\ &\quad - 2E\left[(X_i - X_{i-1})(\overline{X}_{i-1} - X_{i-1})\right].\end{aligned} \tag{12}$$

The last term on the right-hand side of (12) cannot be easily estimated beforehand and should therefore be approximated. We collapse this entire term into a quantity that only depends on the amount of quantization errors D_q from the previous frame, yielding

$$\mathrm{VAR}\left[X_i - \overline{X}_{i-1}\right] = \mathrm{VAR}\left[X_i - X_{i-1}\right] + \kappa D_{\mathrm{q}}(r_{i-1}). \tag{13}$$

We expect the quantization noise of frame X_{i-1} to be only slightly correlated with the frame difference between frames X_{i-1} and X_i. Therefore, we expect the value of κ to be somewhat smaller than one. Note that by combining (13) and (11), D_{q} is defined recursively, thereby making (5) and (6) a dependent optimization problem.

Figure 5 illustrates the relation between the frame difference variance $\mathrm{VAR}[X_1 - \overline{X}_0]$ and the quantization distortion of the first frame of Carphone $\tilde{D}_{\mathrm{q}}$. The first frame is encoded at different bit rates. We observe a roughly linear relation, in this case, with an approximate value of $\kappa = 0.86$.

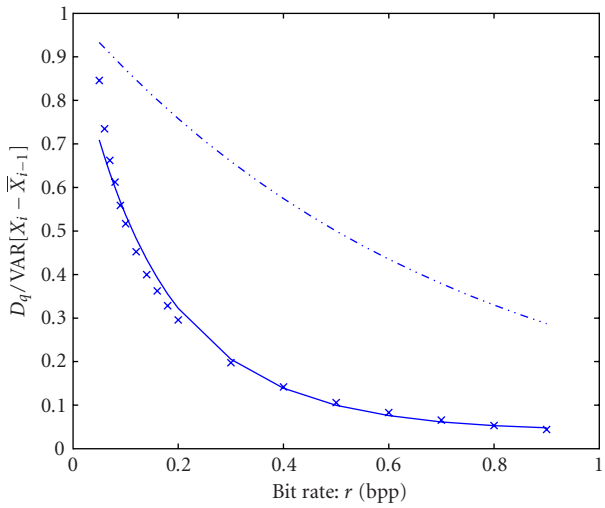


Figure 6: Average RD curve for the first interframe of Carphone. The crosses (×) are the measured normalized distortions $\tilde{D}_q(r_i)$ and the solid line corresponds to the fitted function $2^{g(r)}$. The dashed-dotted line corresponds the RD model 2^{-2r}.

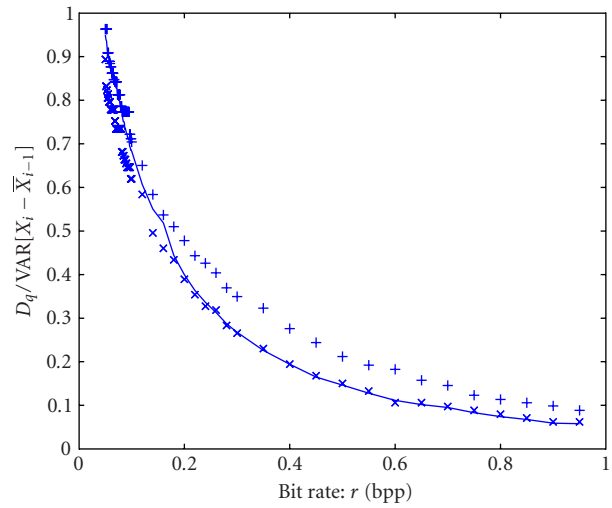


Figure 7: RD curve for the first interframe of Carphone (×), the average RD curve for the first ten frames of Carphone (—), and the RD curve for the first frame of Foreman (+).

We observed similar behavior for other sequences such as Susie and Foreman as well. We therefore postulate that (13) is an acceptable model for calculating the variance $\mathrm{VAR}[X_i - \overline{X}_{i-1}]$ as needed in (11).

The variance $X_i - \overline{X}_{i-1}$ consists of two terms: the quantization distortion of the previous frames, and the frame difference between the current and the previous frame. These two terms might show different RD behavior, that is, a separate $g(r)$ for both terms. However, we assume that both signals show the same behavior since they are both frame-difference signals by nature and not whole frames. The model for predicting the distortion of an interframe now becomes

$$D_q(r_i) = \left(\mathrm{VAR}\left[X_i - X_{i-1}\right] + \kappa D_q(r_{i-1})\right) 2^{g(r_i)}. \tag{14}$$

Figure 6 shows the experimentally obtained RD curve together with a fitted curve representing our model (14). Since this RD curve should not only be valid for varying bit rate r_i but also for varying propagated quantization distortion $D_q(r_{i-1})$, we also vary the bit rate of the previous frame r_{i-1}. Both rates were varied from 0.05 to 0.9. Each value of $D_q(r_i)$ is an average over all settings of r_{i-1}. For completeness, the theoretic curve (8) is shown as well. The function that describes the RD behavior for these frames is

$$D_q(r_i) = \left(\mathrm{VAR}\left[X_i - X_{i-1}\right] + \kappa D_q(r_{i-1})\right) 2^{3.86r_i^2 - 8.15r_i - 0.26}. \tag{15}$$

We then compare the curves for different frames. Figure 7 shows the RD curve for the first frame difference of Carphone and the RD curve for the first frame difference of Foreman as well as the average RD curve for the first ten frame differences of Carphone. This shows again that these curves do not vary much for different video frames and different video sources.

4.3. Channel-induced distortion behavior model

When the channel suffers from high error rates, the channel decoding will not be able to correct all bit errors. Therefore, to solve (5) and (6), we also need a model that describes the behavior of the video decoder in the presence of bitstream errors.

First, we define the channel-induced distortion to be the variance of the difference between the decoded frame ($\overline{X}$) at the encoder side and the decoded frame at the decoder side ($\tilde{X}$):

$$\tilde{D}_e = \mathrm{VAR}[\tilde{X} - \overline{X}]. \tag{16}$$

In [9], a model that describes the coders vulnerability to packet error losses is proposed:

$$D_e = \sigma_{u_0}^2 \,\mathrm{PER}, \tag{17}$$

where $\sigma_{u_0}^2$ is an empirical constant and is found empirically and PER is the packet error rate. Since we are dealing with bit errors and want to predict the impairment on a frame-per-frame basis, we look for a better model.

Modeling the impairments that are due to uncorrected bit errors may result in a detailed analysis of the compression technique used (see, e.g., [13]). Since we desire to have an abstract and a high level model with a limited number of parameters, we base our model on the following three empirical observations.

(1) For both intraframes and interframes, the degree of image impairment due to uncorrected errors depend on the BER. If the individual image impairments caused by channel errors are independent, then the overall effect is the summation of individual impairments. At higher error rates where separate errors cannot be considered independent anymore, we observe

a decreasing influence of the BER. We notice that in a bitstream, a sequence of L bits will be decoded erroneously if one of the bits is incorrect due to a channel error. The probability of any bit being decoded erroneously is then

$$P_{\mathrm{E}}(\mathrm{BER}, L) = 1 - (1 - \mathrm{BER})^{L}. \tag{18}$$

Note that this model describes the behavior related to dependencies between consecutive bits in the bitstream and does not assume any packetization. The value of L is therefore found by curve-fitting and not by an analysis of the data stream structure. Clearly, the value of L will be influenced by the implementation specifics such as resync markers. We interpret L as a value for the effective packet length, that is, the amount of data is lost after a single bit error as if an entire data packet of length L is lost due to an uncorrected error. This model for P_{E} corresponds very well with the observed channel-induced distortion behavior, so we postulate

$$D_{\mathrm{e}} \sim P_{\mathrm{E}} = \left(1 - (1 - \mathrm{BER})^{L}\right), \tag{19}$$

where parameter L was typically found to be in the order of 200 for intraframes and of 1000 for interframes.

(2) For intraframes, the degree of image impairment due to uncorrected errors does not only highly depend on the amount of variance of the original signal but also on the amount of quantization distortion. The expression $\mathrm{VAR}[X_i] - D_{\mathrm{q}}(r_i)$ represents the amount of variance that is encoded; the higher the distortion $D_{\mathrm{q}}(r_i)$, the less information is encoded. We observe that if $D_{\mathrm{q}}(r_i)$ increases, the effect of residual channel errors decreases. Clearly, at $r_i = 0$, nothing is encoded in this frame and the distortion equals the variance. At $r_i \gg 0$, $D_{\mathrm{q}} \approx 0$, there is no quantization distortion, all information is encoded and will be susceptible to bit errors. We therefore postulate

$$D_{\mathrm{e}}(r_i, \mathrm{BER}) \sim \mathrm{VAR}\left[X_i\right] - D_{\mathrm{q}}(r_i). \tag{20}$$

(3) For interframes, we did not observe a statistically significant correlation between the quantization distortion (i.e., the bit rate) and the image impairment due to channel errors. We assume that the image impairment is only related to the variance of the frame difference, thus, here we do not take into account the quantization distortion:

$$D_{\mathrm{e}}(r_i, \mathrm{BER}) \sim \mathrm{VAR}\left[X_i - X_{i-1}\right]. \tag{21}$$

These empirical observations lead us to postulate the following aggregated model of the channel-induced distortions for an intraframe:

$$\begin{aligned} D_{\mathrm{e}}(r_i, \mathrm{BER}) &= \mathrm{VAR}[\tilde{X}_i - \overline{X}_i] \\ &= \alpha P_{\mathrm{E}}(\mathrm{BER}, L_{\mathrm{I}})\left(\mathrm{VAR}\left[X_i\right] - D_{\mathrm{q}}(r_i)\right), \end{aligned} \tag{22}$$

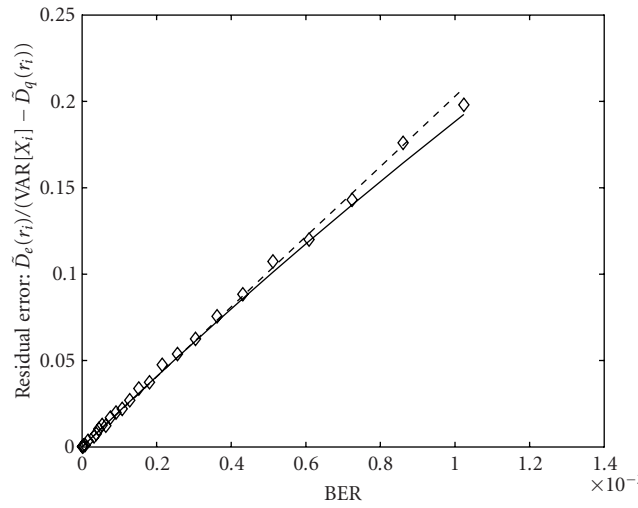


Figure 8: Plot of the normalized distortion $\tilde{D}_{\mathrm{e}}(r_i)/(\mathrm{VAR}[X_i] - \tilde{D}_{\mathrm{q}}(r_i))$ versus BER for the first intraframe of Carphone (shown by $\diamond$). The dashed line corresponds to the simple model $P_{\mathrm{E}} = \mathrm{BER}$ with $\alpha = 255.2$; the solid line to the model $P_{\mathrm{E}} = 1 - (1 - \mathrm{BER})^{202}$ with $\alpha = 1.29$.

and for one interframe:

$$D_{\mathrm{e}}(r_i, \mathrm{BER}) = \beta P_{\mathrm{E}}(\mathrm{BER}, L_{\mathrm{P}})\,\mathrm{VAR}\left[X_i - X_{i-1}\right]. \tag{23}$$

Here, $P_{\mathrm{E}}(\mathrm{BER}, L)$ is given by (18) and L_{I} and L_{P} are the effective packet lengths for intraframes and interframes, respectively. The constants α and β determine to which extent an introduced bit error distorts the picture and need to be found empirically.

For intraframes, $D_{\mathrm{e}}(r_i, \mathrm{BER})$ depends on BER and on the variance $\mathrm{VAR}[X_i] - D_{\mathrm{q}}(r_i)$. Two figures show the curve fitting on this two-dimensional function. Both figures show the results of encoding one frame at different bit rates (ranging from 0.05 to 2.0 bpp) and at different BERs (ranging from 10^{-3} to 10^{-6}), where bit errors were injected in the encoded bitstream randomly. Since we wish to predict the average behavior, we calculated the average distortions of 1000 runs for each setting as follows.

(1) Figure 8 shows the average $\tilde{D}_{\mathrm{e}}$ divided by $\mathrm{VAR}[X_i] - \tilde{D}_{\mathrm{q}}$ as a function of BER. The dashed line corresponds to a line fitted with $P_{\mathrm{E}} = \mathrm{BER}$ and $\alpha = 255.2$. We observe that it deviates at higher BER. The solid line corresponds to $P_{\mathrm{E}} = 1 - (1 - \mathrm{BER})^{L_{\mathrm{I}}}$ with an effective packet length $L_{\mathrm{I}} = 202$ and $\alpha = 1.29$, which gives a better fit.

(2) Figure 9 shows $\tilde{D}_{\mathrm{e}}$ divided by $P_{\mathrm{E}}(\mathrm{BER}, L_{\mathrm{I}} = 202)$ as a function of $\mathrm{VAR}[X_i] - \tilde{D}_{\mathrm{q}}$. The fitted line crosses the origin. Clearly, this model does not fit these measurements extremely well because the effect of $D_{\mathrm{q}}(r_i)$ is very unpredictable. On the other hand, because the model catches the coarse behavior, we still can incorporate the effect that $D_{\mathrm{q}}(r_i)$ has on the channel-induced distortion. For other sources (Foreman, Susie), we observe a similar behavior.

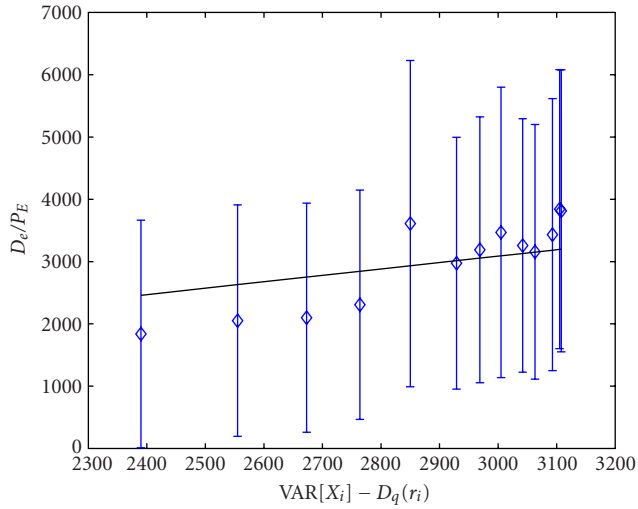


FIGURE 9: Plot of $\mathrm{VAR}[X_i] - \tilde{D}_q(r_i)$ versus the normalized distortion $\tilde{D}_e(r_i, \mathrm{BER})/P_E(\mathrm{BER}, L_I)$ (shown by $\diamond$) for the first intraframe of Carphone. The error bars represent the standard deviation over 1000 runs of the experiment. The solid line represents our model $D_e(r_i, \mathrm{BER})/P_E(\mathrm{BER}, L_I) = \alpha(\mathrm{VAR}[X_i] - D_q(r_i))$.

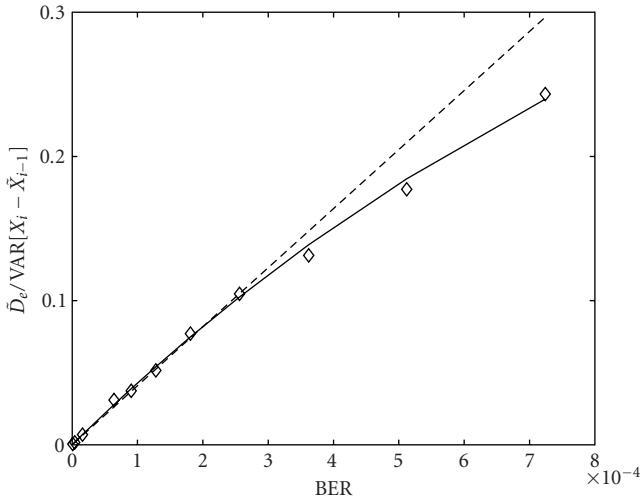


FIGURE 10: Plot of the normalized channel-induced distortion $\tilde{D}_e(r_i, \mathrm{BER})/\mathrm{VAR}[X_i - X_{i-1}]$ versus BER (shown by $\diamond$). The values are averaged over the first ten interframes of Carphone. The dashed line corresponds to the model $P_E = \mathrm{BER}$, and the solid line corresponds to the model $P_E = 1 - (1 - \mathrm{BER})^{876}$ with $\alpha = 0.51$.

Finally, for interframes, $D_e(r_i, \mathrm{BER})$ only depends on BER and on the constant factor $\mathrm{VAR}[X_i - X_{i-1}]$. Figure 10 shows the average $\tilde{D}_e$ divided by $\mathrm{VAR}[X_i - X_{i-1}]$ versus the BER. The resulting curve corresponds to $P_E = 1 - (1 - \mathrm{BER})^{L_P}$ with $L_P = 876$. Here, we found $\beta = 0.51$.

4.3.1. Error propagation in interframes

Due to the recursive structure of the interframe coder, decoding errors introduced in a frame will cause temporal error propagation [9, 14]. Since (5) and (6) tries to minimize the distortion over a whole GOP, we have to take this propagation into account for each frame individually. In [9], a high-level model was proposed to describe the error propagation in motion-compensated DCT-based video encoders including a loop filter. We adopted the λ factor which describes an exponential decay of the propagated error, but we discarded the γ factor which models propagation of errors in motion-compensated video, yielding

$$D_e(r_i, \mathrm{BER}) = (1-\lambda) D_e(r_{i-1}, \mathrm{BER}) + \beta\left(1 - (1 - \mathrm{BER})^{L_P}\right) \mathrm{VAR}[X_i - X_{i-1}]. \tag{24}$$

Our observations are that this is an accurate model although the propagated errors decay only slightly. For instance, for the Carphone sequence, we found that $\lambda = 0.02$ (not shown here). In a coder where loop filtering is used to combat error propagation, this factor is much higher [9].

5. MODEL VALIDATION

We have now defined all models needed to solve (5) and (6). Assuming we know the variances $\mathrm{VAR}[X_i]$, $\mathrm{VAR}[X_i - X_{i-1}]$, the parameters for the functions $f(r)$, $g(r)$, and the model parameters κ, L_I, L_P, α, and β, we can minimize (5) and (6) using these models. Note that since in principle each frame can have its own RD function, the function will get the additional parameter i to signify that

$$\begin{aligned}
D_{\mathrm{GOP}} &= \frac{1}{N}\sum_{i=0}^{N-1}\{D_q(r_i|i) + D_e(r_i, \mathrm{BER}\,|i)\},\\
D_q(r_0|i=0) &= \mathrm{VAR}[X_0]\, 2^{f(r_0|0)}, \quad \text{for } i = 0,\\
D_e(r_0, \mathrm{BER}\,|i=0) &= \alpha\left(1 - (1-\mathrm{BER})^{L_I}\right)\left(\mathrm{VAR}[X_0] - D_q(r_0|0)\right), \quad \text{for } i > 0,\\
D_q(r_i|i) &= \left(\mathrm{VAR}[X_i - X_{i-1}] + \kappa D_q(r_{i-1}|i-1)\right) 2^{g(r_i|i)}, \quad \text{for } i > 0,\\
D_e(r_i, \mathrm{BER}\,|i) &= (1-\lambda) D_e(r_{i-1}, \mathrm{BER}\,|i) + \beta\left(1 - (1-\mathrm{BER})^{L_P}\right)\mathrm{VAR}[X_i - X_{i-1}], \quad \text{for } i > 0.
\end{aligned} \tag{25}$$

In this section, we will verify these models by encoding a sequence of frames with different bit rate allocations and compare the measured distortion and the predicted distortion. Furthermore, we will introduce bit errors in the bitstream and verify the prediction of the distortion under error prone channel conditions. As mentioned in the introduction, in this paper, we do not optimize (5) and (6) using the models (25)—as would be required in a real-time implementation. Instead, we aim to show that it is possible to predict the overall distortion for a GOP under a wide range of channel conditions. We will show that a setting for N and r_i optimized with our behavior models (25) indeed yields a solution that is close to the measured minimum.

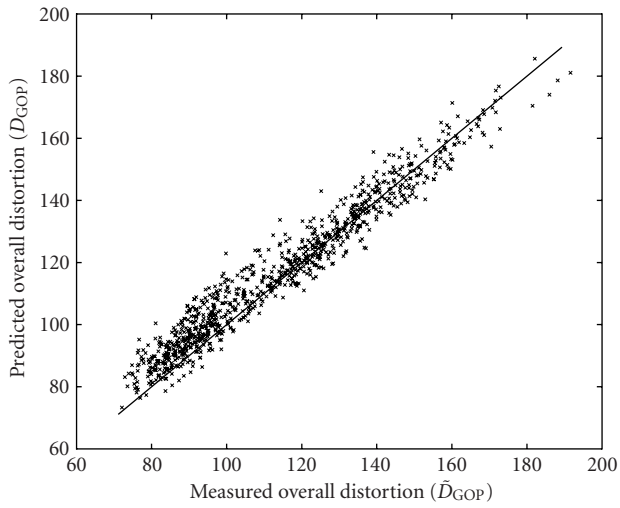


FIGURE 11: For each possible bit rate assignment, the cross (×) shows the measured distortion $\tilde{D}_{GOP}$ horizontally and the predicted distortion D_{GOP} vertically. The line represents the points where the measurements would match the predicted distortion.

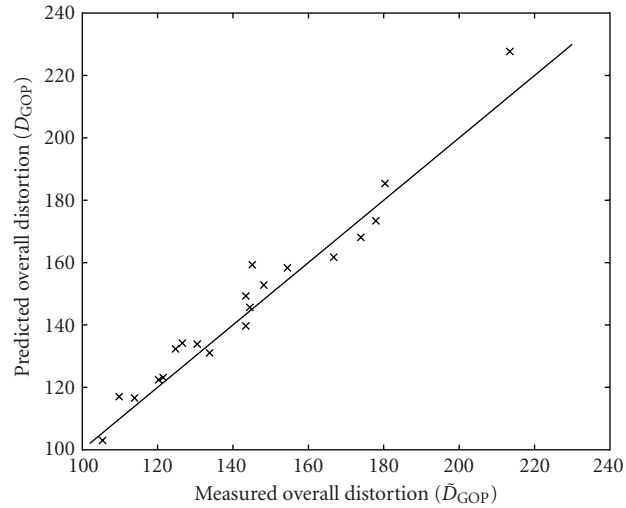


FIGURE 12: Selection of 20 bit rate assignments when BER = $32 \cdot 10^{-6}$. For each case the cross (×) shows the measured distortion $\tilde{D}_{GOP}$ horizontally and the predicted distortion D_{GOP} vertically. The solid line represents the points where the predicted distortion and the measured distortion would match.

To validate our model, we will compare the measurements of the overall distortion of a GOP with the predictions made with our model (25). We used the JPEG2000 encoder/decoder as our video coder (Figure 2), and encoded the Carphone sequence. In the first experiment, a GOP of ten frames was encoded with different bit rate allocations. No residual channel errors are introduced. In the second experiment, random bit errors were introduced in the encoded bitstream to simulate an error prone channel. In the third experiment, we addressed the issue of finding the optimal GOP length. In all these experiments, we used the models (25) and the parameters we have obtained in Section 4 for the first ten frames of Carphone. In the last experiment, we used our models to optimize the settings for a whole sequence. We compare optimizing the settings with our models and with two other simple rate allocations. Furthermore, we have investigated the gain that can be achieved if the RD curves are known for each individual frame instead of the average RD curves.

5.1. Optimal rate allocation

In this experiment, no residual channel errors were present (BER = 0) and the average bit rate available for each frame was 0.2 bpp. To each frame, we assigned bit rates varying from 0.1, 0.2, 0.3 to 1.1 bpp, while keeping the average bit rate constant at 0.2 bpp. The GOP length was set to 10. The total number of possible bit rate allocations with these constraints is 92378.

A GOP of ten frames was encoded with each of these bit rate allocations. We then measured the overall distortion denoted by $\tilde{D}_{GOP}$ and compared that with the predicted distortion D_{GOP} (using (4), (10), and (15)). Figure 11 shows the results. All points were plotted with the measured distortion $\tilde{D}_{GOP}$ on the horizontal axis. The vertical axis shows the predicted distortion D_{GOP}. The straight line corresponds to the points where the prediction matches the measured values. Points under this line underestimate the measured overall distortion and the points above the line overestimate the measured overall distortion. The region we are interested in is located in the lower left area where the bottom-most point represents the bit rate allocation that minimizes our model, D_{GOP} (25). The cloud shape gives good insight in the predictive strength of the model since the points are never far off the corresponding measured distortion.

As we can see in Figure 11, the predicted distortion and the measured distortion correspond well over the whole range of bit rate allocations. Note that although it is not possible with these proposed behavior models to find the exact values of r_i yielding the minimal measured distortion (we only know the exact distortion after encoding and decoding), the predicted minimal distortion is close to the measured minimum distortion. We use the following metrics to express the performance of the model: the relative error

$$\varepsilon_1 = \mathrm{E}\left[\frac{D_{GOP} - \tilde{D}_{GOP}}{\tilde{D}_{GOP}}\right] \cdot 100\%, \tag{26}$$

and the standard deviation of the relative error:

$$\varepsilon_2 = \mathrm{std}\left[\frac{D_{GOP} - \tilde{D}_{GOP}}{\tilde{D}_{GOP}}\right] \cdot 100\%. \tag{27}$$

For this experiment, $\varepsilon_1 = 3.2\%$, which means that we slightly overestimated all distortions; $\varepsilon_2 = 5.7\%$, which means that on average our predictions were within $3.2 - 5.7 = -2.5\%$ and $3.2 + 5.7 = 8.9\%$ around the measured values.

We can interpret this in terms of PSNR: an increase of the error variance of 5.7% corresponds to a decrease of the PSNR by $10 \log 1.089 = 0.37$ dB. This means that we predicted the average quality with 0.37 dB accuracy.

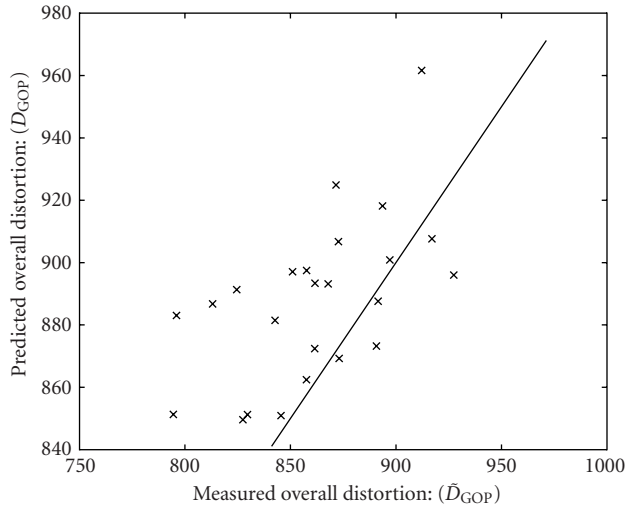


FIGURE 13: Selection of 20 bit rate assignments when BER = $1024 \cdot 10^{-6}$. For each case, the cross ($\times$) shows the measured distortion $\tilde{D}_{GOP}$ horizontally and the predicted distortion D_{GOP} vertically. The solid line represents the points where the predicted distortion and the measured distortion would match.

5.2. Optimal rate allocation for a channel with residual errors

When residual channel errors were introduced, the same experiment yielded different results at different runs because of the randomness of bit errors. Therefore, for each rate allocation, the coding should be done at least a thousand times and the measured distortion values should be averaged. Analyzing each bit allocation with such accuracy is very demanding in terms of computing time, therefore, we selected twenty cases uniformly distributed from the 92378 rate allocations to gain sufficient insight in the predictive power of the behavior models.

For this experiment, we chose BER = $32 \cdot 10^{-6}$. Figure 12 shows the measured average distortion $\tilde{D}_{GOP}$ and the predicted distortion D_{GOP} for the 10-frame case. Now, the relative error is $\varepsilon_1 = 2.0\%$ and $\varepsilon_2 = 3.7\%$.

Note that in these simulations, we did not use any special settings of a specific video coder and we used no error concealment techniques other than the standard JPEG2000 error resilience. Because of the combination of wavelet transforms and progressive bit plane coding in JPEG2000, in most cases the bit errors only caused minor distortions in the higher spatial frequencies. However, sometimes a lower spatial frequency coefficient was destroyed yielding a higher distortion.

Any individual random distortion can differ greatly from the predicted one. Because large distortions are less likely to occur than small distortions, our model gives a boundary on the resulting distortion. We measured that for 88.0% of the cases, the measured distortion was lower than the predicted value.

We then changed our BER to $1024 \cdot 10^{-6}$. Figure 13 shows the measured and the predicted distortions. For this high BER, the relative performance metrics were still good, $\varepsilon_1 = 0.31\%$ and $\varepsilon_2 = 3.6\%$. Note that these relative metrics are similar to the case without channel errors. This means that on the average, although the channel-error distortion is hard to predict, our model is still able to make good predictions of the average distortion even under error prone conditions. Apparently, the average D_e part of the total distortion is very predictable, this is probably due to the good error resilience of the JPEG2000 encoder we used.

5.3. Selection of the optimal GOP length

In the previous experiments, the optimal bit rate allocation was selected for each frame. This experiment deals with selecting the optimal GOP length N. The same constraints were used as in the previous experiment, but now the GOP length varied from 1 to 10.

Figure 14 shows for each GOP length from 1 to 10 the bit rate allocations for BER = 0. Observe that the average bit rate of 0.2 bpp per frame is spread out over each frame in the GOP to obtain a minimal overall distortion D_{GOP}. The last case ($N = 10$) corresponds to the bottom-most point in Figure 11.

Figure 15 shows the predicted overall distortion D_{GOP} and measured overall distortion $\tilde{D}_{GOP}$ for each of these bit rate allocations. Following our criterion (5) and (6), the optimal GOP length is $N = 8$. Since interframes are used, we expect that using larger GOPs gives lower distortions. This is generally true, but in these experiments we did not cover the whole solution space since we used increments of 0.1 bpp for the bit rates. With this limited resolution, we may find suboptimal solutions.

Figure 16 shows the result of a simulation where N varied from 1 to 15. In this simulation we only used our models to predict the distortion; the corresponding measurements were not carried out due to computational limitations (there are 600 000 combinations of rate allocations when bit rates $r_i \in \{0.1, 0.2, \ldots, 1.6\}$ are used). The distortions were again minimized with an average bit rate constraint of 0.2 bpp. The points correspond to the minimum achievable distortion D_{GOP} at each GOP length. We see that for $N > 6$, the average distortion did not substantially decrease anymore, so larger GOP lengths would not improve the quality greatly. Figure 16 also shows the results of the simulations for BER = $\{32 \cdot 10^{-6}, 256 \cdot 10^{-6}, 512 \cdot 10^{-6}\}$. Note that at some point, the accumulated channel-induced distortion becomes higher than the gain we obtain from adding another interframe. At this point, the internal controller should decide to encode a new intraframe to stop the error propagation.

5.4. Optimal rate allocation for whole sequences

In this experiment, we used our models and our optimization criterion to optimize the settings for the whole sequence of Carphone.

We have compared the measured distortion with two other simple rate allocation methods.

(1) The rates and GOP length settings are obtained using our models and optimization criterion with the

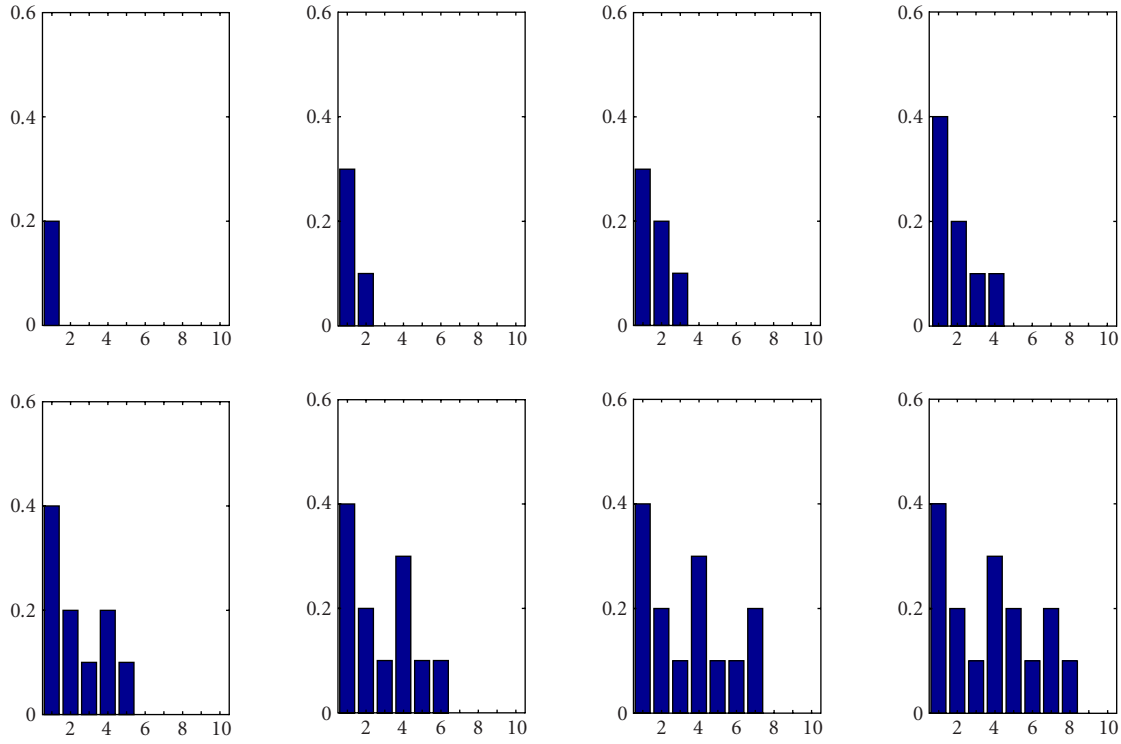


Figure 14: Bit rate allocations for BER = 0. Every plot corresponds to a GOP length running from $N = 1$ to 10. Within each plot, for each frame, the bit rate allocation that minimizes D_{GOP} is shown and the average bit rate of r is 0.2 bpp.

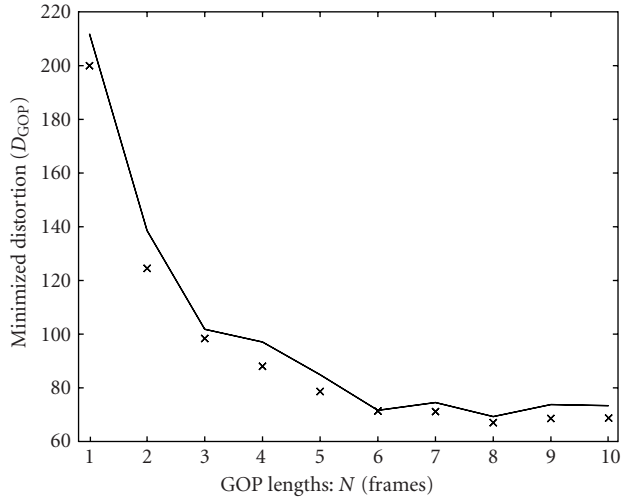


Figure 15: Minimized distortion D_{GOP} (—) and $\tilde{D}_{\text{GOP}}$ (×) for GOP lengths between 1 and 10 and for an average bit rate of $r = 0.2$ bpp.

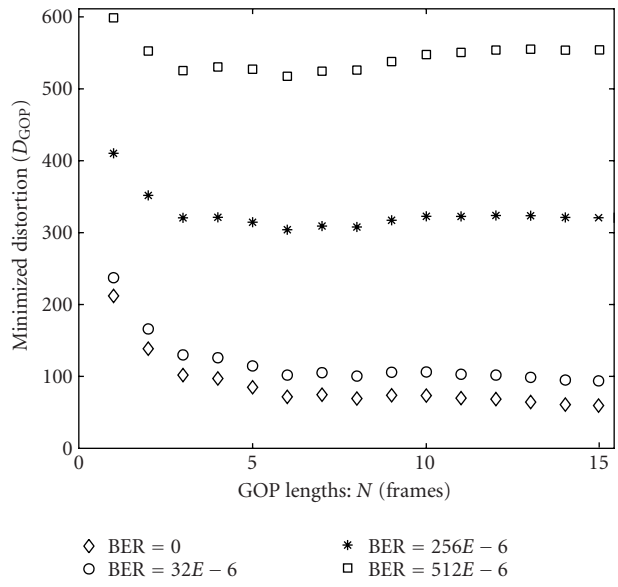


Figure 16: Minimized distortion D_{GOP} for GOP lengths between 1 and 15, for different BERs, and an average bitrate of $r = 0.2$ bpp.

constraints that $N_{\max} = 10$ and the average bit rate is 0.2.

(2) Every frame has the same fixed bit rate $r = 0.2$. The GOP length is obtained using our models and optimization criterion.

(3) Every frame has the same fixed bit rate $r = 0.2$. The GOP length has a fixed value of 10.

These methods were applied to the Carphone and the Susie sequences for BER = 0, BER = $128 \cdot 10^{-6}$, and BER = $512 \cdot 10^{-6}$. The results are shown in Table 1. For Carphone, method (1) is clearly better than method (3). Method (2) and

Table 1: Comparison between different rate allocation methods.

Case	Method	Distortion		
		1	2	3
Carphone	BER = 0	76.6	91.1	90.6
Carphone	BER = $128 \cdot 10^{-6}$	136.8	161.3	161.1
Carphone	BER = $512 \cdot 10^{-6}$	397.7	408.4	410.4
Susie	BER = 0	28.6	28.9	28.9
Susie	BER = $128 \cdot 10^{-6}$	47.4	49.6	59.5
Susie	BER = $512 \cdot 10^{-6}$	116.4	117.1	151.2

method (3) perform more or less the same. When bit errors are introduced, method (1) still outperforms the other two. For Susie, method (1) also outperforms the other two. When bit errors are present, method (2) (just adapting the GOP length) greatly outperforms method (3). We conclude that the performance of our method depends heavily on whether the characteristics of the source are changing over time or not. It seems that either optimizing the GOP length or the bit rates decreases the distortion as opposed to method (3).

Finally, we have investigated whether using RD parameters for each individual frame instead of average RD parameters, indeed gives a significant increase of the performance. We compared the case where for each individual frame the corresponding RD function is used for optimization (case 1), and the case where one average RD function is used for the whole sequence (case 2). For Carphone, we measured the following: for case 1, the average distortion $D = 76.5$, for case 2, $D = 91.0$. This means that significant gains can be expected when the RD curves are known for each frame. Of course in practice this is not possible. On the other hand, since consecutive frames look alike, we believe that an adaptive method to obtain the RD curves from previous frames could give significant gains. For Susie we have similar results. For case 1, $D = 28.6$, and for case 2, $D = 47.9$.

6. DISCUSSION

In this paper, we introduced a behavior model that predicts the overall distortion of a group of pictures. It incorporates the structure and prediction scheme of most video coders to predict the overall distortion on a frame-per-frame basis. Furthermore, the model corrects for statistical dependencies between successive frames. Finally, our model provides a way to predict the channel-induced distortion when residual channel errors are present in the transmitted bit steam.

Although the deviation of the model predicted distortion from the measured distortion can become substantial, with this model we can still compare different settings and select one likely to cause the smallest distortion.

Our models are designed to closely follow the behavior of the encoder, given the characteristics of the video data, and to make an accurate prediction of the distortion for each frame. These predictions are made before the actual encoding of the entire group of pictures. To predict the average distortion, we need to know the variance of each frame and the variance of the frame difference of the consecutive original frames. We also need two parameterized rate distortion curves and six other parameters (κ, α, β, L_I, L_P, and λ).

In our experiments—some of which were shown in this paper—we noticed that these parameters do not change greatly between consecutive group of pictures, therefore they can be predicted recursively from the previous frames that have already been encoded. On the other hand, we have shown that significant gains can be expected when the rate distortion parameters are obtained adaptively and no average rate distortion curves are used. The factors κ, α, β, L_I, L_P, and λ do not depend greatly on the source data, but rather on the coder design, and thus may be fixed for a given video encoder.

After obtaining the frame differences, the distortion can be predicted before the actual encoding takes place. This makes the model suitable for rate control and constant bit rate coding as well as for quality of service controlled encoders. Although this paper focused on rate allocation of entire frames rather than on macroblocks, all models can be generalized for use at the macroblock level.

REFERENCES

[1] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*, International Thompson Publishing, London, UK, 1996.

[2] G. Côté, S. Shirani, and F. Kossentini, "Optimal mode selection and synchronization for robust video communications over error prone networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 952–968, 2000.

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

[4] M. Brystrom and J. W. Modestino, "Combined source channel coding for transmission of video over a slow-fading rician channel," in *Proc. International Conference on Image Processing*, vol. 2, pp. 147–151, Chicago, Ill, 1998.

[5] K. Stuhlmüller, N. Färber, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1012–1032, 2000.

[6] A. van der Schaaf and R. L. Lagendijk, "Independence of source and channel coding for progressive image and video data in mobile communications," in *Proc. Visual Communications and Image Processing*, vol. 4067, pp. 187–197, Perth, Australia, June 2000.

[7] H. van Dijk, K. Langendoen, and H. Sips, "ARC: a bottom-up approach to negotiated QoS," in *Proc. 3rd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 128–137, Monterey, Calif, USA, December 2000.

[8] Y. S. Chan and J. W. Modestino, "Transport of scalable video over CDMA wireless networks: a joint source coding and power control approach," in *Proc. International Conference on Image Processing*, vol. 2, pp. 973–976, Thessaloniki, Greece, October 2001.

[9] N. Färber, K. Stuhlmüller, and B. Girod, "Analysis of error propagation in hybrid video coding with application to error resilience," in *Proc. International Conference on Image Processing*, vol. 2, pp. 550–554, Kobe, Japan, 1999.

[10] J. R. Taal, K. Langendoen, A. van der Schaaf, H. W. van Dijk, and R. L. Lagendijk, "Adaptive end-to-end optimization of mobile video streaming using QoS negotiation," in *Proc. International Symposium on Circuits and Systems*, vol. 1, pp. 53–56, Scottsdale, Ariz, USA, May 2002.
[11] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 533–545, 1994.
[12] M. Boliek et al., "Jpeg 2000 part 1 final committee draft, version 1.0," Tech. Rep., JPEG, 2002.
[13] G. Reyes, A. R. Reibman, and S. F. Chang, "A corruption model for motion compensated video subjected to bit errors," in *Proc. Packet Video Workshop '99*, NY, USA, April 1999.
[14] J. G. Kim, J. Kim, and C. C. J. Kuo, "Corruption model of loss propagation for relative prioritized packet video," in *Proc. SPIE Applications of Digital Image Processing XXIII*, vol. 4115, pp. 214–224, San Diego, July 2000.

Jacco R. Taal received his M.S. degree in Electrical Engineering from Delft University of Technology, Delft, The Netherlands, in 2001. At present he is pursuing his Ph.D. degree at the same university. His research interests include real-time video compression for wireless communications and peer-to-peer systems. Currently he is doing research on video transmissions for peer-to-peer communications and ad hoc networks.

Zhibo Chen received his B.S., M.S., and Ph.D. degrees from the Department of Electrical Engineering, Tsinghua University, Beijing, China, in 1998, 2000, and 2003, respectively. He is currently with Sony Research Center, Tokyo. His research interests include video coding theory and algorithm and video communication over networks.

Yun He received the B.S. degree in signal processing from Harbin Shipbuilding Institute, Harbin, China, in 1982, the M.S. degree in ultrasonic signal processing from Shanghai Jiaotong University, Shanghai, China in 1984, and the Ph.D. degree in image processing from Liege University, Liege, Belgium, in 1989. She is currently an Associate Professor at Tsinghua University, Beijing, China. She serves as a Senior Member in IEEE, Technical Committee Member of Visual Signal Processing and Communications in IEEE CAS Society, Picture Coding Symposium Steering Committee Member, as a Program Committee Member in SPIE Conference of Visual Communications and Image Processing (2000-2001). Her research interests include picture coding theory and methodology, picture coding algorithm software and hardware complexity analysis, video codec VLSI structure, and multiview and 3D picture coding.

R. (Inald) L. Lagendijk received his M.S. and Ph.D. degrees in electrical engineering from Delft University of Technology in 1985 and 1990, respectively. Since 1999, he has been a Full Professor in the Information and Communication Theory Group of Delft University of Technology. Prof. Lagendijk was a visiting scientist at Eastman Kodak Research (Rochester, NY) in 1991 and a Visiting Professor at Microsoft Research and Tsinghua University, Beijing, China, in 2000 and 2003. Prof. Lagendijk is the author of the book *Iterative Identification and Restoration of Images* (Kluwer, 1991) and coauthor of the books *Motion Analysis and Image Sequence Processing* (Kluwer, 1993) and *Image and Video Databases: Restoration, Watermarking, and Retrieval* (Elsevier, 2000). He has served as an Associate Editor of the *IEEE Transactions on Image Processing*, and he is currently an Associate Editor of the IEEE Transactions on Signal Processing Supplement on Secure Digital Media, and an Area Editor of Eurasip journal Signal Processing: Image Communication. At present his research interests include signal processing and communication theory, with emphasis on visual communications, compression, analysis, searching, and watermarking of image sequences. He is currently leading and actively involved in a number of projects in the field of data hiding and compression for multimedia communications.

EURASIP Journal on Applied Signal Processing 2004:2, 304–316

An Integrated Source and Channel Rate Allocation Scheme for Robust Video Coding and Transmission over Wireless Channels

Jie Song
Media Connectivity Division, Agere Systems, Holmdel, NJ 07733, USA
Email: jiesong@agere.com

K. J. Ray Liu
Electrical & Computer Engineering Department, University of Maryland, College Park, MD 20742, USA
Email: kjrliu@eng.umd.edu

Received 22 November 2002; Revised 3 September 2003

A new integrated framework for source and channel rate allocation is presented for video coding and transmission over wireless channels without feedback channels available. For a fixed total channel bit rate and a finite number of channel coding rates, the proposed scheme can obtain the near-optimal source and channel coding pair and corresponding robust video coding scheme such that the expected end-to-end distortion of video signals can be minimized. With the assumption that the encoder has the stochastic information such as average SNR and Doppler frequency of the wireless channel, the proposed scheme takes into account robust video coding, channel coding, packetization, and error concealment techniques altogether. An improved method is proposed to recursively estimate the end-to-end distortion of video coding for transmission over error-prone channels. The proposed estimation is about 1–3 dB more accurate compared to the existing integer-pel-based method. Rate-distortion-optimized video coding is employed for the trade-off between coding efficiency and robustness to transmission errors.

Keywords and phrases: multimedia communications, joint source and channel coding, wireless video.

1. INTRODUCTION

Multimedia applications such as video phone and video streaming will soon be available in the third generation (3G) wireless systems and beyond. For these applications, delay constraint makes the conventional automatic repeat request (ARQ) and the deep interleaver not suitable. Feedback channels can be used to deal with the error effects incurred in image and video transmission over error-prone channels [1], but in applications such as broadcasting services, there is no feedback channel available. In such cases, the optimal trade-off between source and channel coding rate allocations for video transmission over error-prone channels becomes very important. According to Shannon's separation theory, these components can be designed independently without loss in performance [2]. However, this is based on the assumption that the system has an unlimited computational complexity and infinite delay. These assumptions are not satisfied in delay-sensitive real-time multimedia communications. Therefore, it is expected that joint considerations of source and channel coding can provide performance improvement [3, 4].

Most of the joint source and channel coding (JSCC) schemes have been focusing on images and sources with ideal signal models [4, 5]. For video coding and transmission, many works still keep the source coding and channel coding separate instead of optimizing their parameters jointly from an overall end-to-end transmission point of view [6, 7]. Some excellent reviews about robust video coding and transmission over wireless channels can be found in [8, 9]. In [10], a JSCC approach is proposed for layered video coding and transport over error-prone packet networks. It presented a framework which trades video source coding efficiency off for increased bitstream error resilience to optimize the video coding mode selection with the consideration of channel conditions as well as error recovery and concealment capabilities of the channel codec and source decoder, respectively. However, the optimal source and channel rate allocation and corresponding video macroblock (MB) mode selection have to be selected through simulations over packet-loss channel models. In [11], a parameterized model is used for the analysis of the overall mean square error (MSE) in hybrid video coding for the error-prone transmission. Models for the video encoder, a bursty transmission channel, and error

propagation at the video decoder have been combined into a complete model of the entire video transmission system. However, the model for video encoder involves several parameters and the model is not theoretically optimal because of the use of random MB intramode updating, which does not consider the different motion activities within a video frame to deal with error propagation. Furthermore, the models depend on the distortion-parameter functions obtained through ad hoc numerical models and simulations over specific video sequences, which also involves a lot of simulation efforts and approximation. The authors of [12] proposed an operational rate-distortion (RD) model for DCT-based video coding incorporating the MB intra–refreshing rate and an analytic model for video error propagation which has relatively low computational complexity and is suitable for real-time wireless video applications. Both methods in [11, 12] focus on the statistical model optimization for general video sequence, which is not necessarily optimal for a specific video sequence because of the nonstationary behavior across different video sequences.

In this paper, we propose an integrated framework to obtain, the near-optimal source and channel rate allocation, and the corresponding robust video coding scheme for a given total channel bit rate with the knowledge of the stochastic characteristics of the wireless fading channel. We consider the video coding error (quantization and mode selection of MB), error propagation, and concealment effects at the receiver due to transmission error, packetization, and channel coding in an integrated manner. The contributions of this paper are the following. First, we present an integrated system design method for wireless video communications in realistic scenarios. This proposed method takes into account the interactions of fading channel, channel coding and packetization, and robust video coding in an integrated, yet simple way, which is an important system design issue for wireless video applications. Second, we propose an improved video distortion estimation which is about 1–3 dB peak signal-to-noise ratio (PSNR) more accurate than the original integer-pel-based method (IP) in [13] for half-pel-based video coding (HP), and the computational complexity in the proposed method is less than that in [13].

The rest of the paper is organized as follows. Section 2 describes first the system to be studied, then the packetization and channel coding schemes used. We also derive the integrated relation between MB error probability and channel coding error probability given the general wireless fading channel information such as average signal-to-noise ratio (SNR) and Doppler frequency. Section 3 presents the improved end-to-end distortion estimation method for HP-based video coding. Simulations are performed to compare the proposed method to the IP-based method in [13]. Then we employ RD-optimized video coding scheme to optimize the end-to-end performance for each pair of source and channel rate allocation. Simulation results are shown in Section 4 to demonstrate the accuracy of the proposed end-to-end distortion estimation algorithm under different channel characteristics. Conclusions are stated in Section 5.

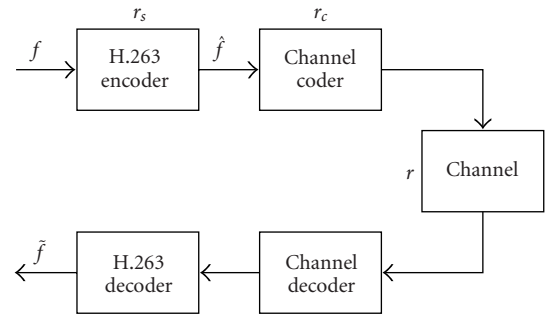


Figure 1: Joint source and channel video coding.

2. PROBLEM DEFINITION AND INTEGRATED SYSTEM STRUCTURE

The problem to be studied is illustrated in Figure 1 which can be specified by five parameters (r, r_c, ρ, f_d, F): r is the total channel bit rate, r_c is the channel coding rate, ρ is the average SNR at the receiver, f_d is the Doppler frequency of the fading channel targeted, and F is the video frame rate. H.263 [14] is used for video coding. A video sequence denoted as $f_l^{\mathbf{s}}$, where $\mathbf{s} = (x, y)$, $1 \le x \le X$, $1 \le y \le Y$, is the pixel spatial location and $l = 1, \ldots, L$ is the frame index, is encoded at the bit rate $r_s = r \times r_c$ b/s and the frame rate F f/s with the MB error probability $P_{Mb} = f(\rho, f_d, r_c)$ that will be detailed next. The resulted H.263 bitstream is packetized and protected by forward error correction (FEC) channel coding with the coding rate r_c. The resulted bitstream with rate r b/s is transmitted through wireless channels characterized by ρ and f_d. The receiver receives the bitstream corrupted by the channel impairment, then reconstructs the video sequence $\tilde{f}_l^{\mathbf{s}}$ after channel decoding, H.263 video decoding, and possible error concealment if residual errors occur. The end-to-end MSE between the input video sequence at the encoder and the reconstructed video sequence at the decoder is defined as

$$D_E(r_s, r_c) = \frac{1}{XYL} \sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{l=1}^{L} E\left\{ \left[f_l^{(x,y)} - \tilde{f}_l^{(x,y)}(r_s, r_c) \right]^2 \right\}. \tag{1}$$

For the video system in Figure 1, there are two tasks to be performed with the five given system parameters (r, r_c, ρ, f_d, F). First, we need to decide how to allocate the total fixed bit rate r to the source rate $r_s = r \times r_c$ to minimize the end-to-end MSE of the video sequence. Furthermore, the video encoder should be able to, for a source/channel rate allocation (r_s, r_c) with residual channel decoding failure rate denoted as $p_w(r_c)$, select the coding mode and quantizer for each MB to minimize the end-to-end MSE of the video sequence. The goal is to obtain the source/channel rate pair (r_s^*, r_c^*) *and* the corresponding robust video coding scheme to minimize (1).

In practical applications, there are only finite number of source/channel pairs available. We can find the robust video encoding schemes for each rate pair (r_s, r_c) that minimizes (1) and denote the minimal end-to-end MSE obtained as

$D_E^*(r_s, r_c)$, then the optimal source/channel rate pair (r_s^*, r_c^*) and the corresponding video coding scheme can be obtained as

$$(r_s^*, r_c^*) = \underset{(r_s, r_c)}{\operatorname{argmin}} D_E^*(r_s, r_c). \tag{2}$$

For each pair (r_s, r_c), we use RD-optimized video coding scheme to trade off between the source coding efficiency and robustness to error propagation. An improved recursive method which takes into account the interframe prediction, error propagation, and concealment effect is used to estimate the end-to-end MSE frame by frame. In this paper, the wireless fading channel is modeled as a finite-state Markov chain (FSMC) model [15, 16, 17], and the Reed-Solomon (RS) code is employed for forward error coding.

2.1. Modeling fading channels using finite-state Markov chain

Gilbert and Elliott [15, 16] studied a two-state Markov channel model, where each state corresponds to a specific channel quality. This model provides a close approximation for the error rate performance of block codes on some noisy channels. On the other hand, when the channel quality varies dramatically such as in a fast Doppler spread, the two-state Gilbert-Elliott model becomes inadequate. Wang and Moayeri extended the two-state model to an FSMC model for characterizing the Rayleigh fading channels [17]. In [17], the received SNR is partitioned into a finite number of intervals. Denote by $0 = A_0 < A_1 < A_2 < \cdots < A_K = \infty$ the SNR thresholds of different intervals, then if the received SNR is in the interval $[A_k, A_{k+1})$, $k \in \{0, 1, 2, \ldots, K-1\}$, the fading channel is said to be in state S_k. It turns out that if the channel changes slowly and is properly partitioned, each state can be considered as a steady state, and a state transition can only happen between neighboring states. As a result, a fading channel can be represented using a Markov model if given the average SNR ρ and Doppler frequency f_d.

2.2. Performance analysis of RS code over finite-state Markov channel model

RS codes possess maximal minimum distance properties which make them powerful in correcting errors with arbitrary distributions. For RS symbols composed of m bits, the encoder for an RS(n, k) code groups the incoming bitstream into blocks of k information symbols and appends $n-k$ redundancy symbols to each block. So the channel coding rate is $r_c = k/n$. For an RS(n, k) code, the maximal number of symbol errors that can be corrected is $t = \lfloor (n-k)/2 \rfloor$. When the number of symbol errors is more than t, RS decoder reports a flag to notify that the errors are uncorrectable. The probability that a block cannot be corrected by RS(n, k), denoted as a decoding failure probability $p_w(n, k)$, can be calculated as

$$p_w(n, k) = \sum_{m=t+1}^{n} P(n, m), \tag{3}$$

where $P(n, m)$ denotes the probability of m symbol errors within a block of n successive symbols. The computation of $P(n, m)$ for FSMC channel model has been studied before (see [16, 18]).

2.3. Packetization and macroblock error probability computation

We use baseline H.263 video coding standard for illustration. H.263 GOB/slice structure is used where each GOB/slice is encoded independently with a header to improve resynchronization. Denoting by N_s the number of GOB/slice in each frame, the RS(n, k) code block size n (bytes) is set to

$$n = \left\lceil \frac{r}{8 \cdot F \cdot N_s} \right\rceil \tag{4}$$

such that each GOB/slice is protected by an RS codeword in average, where $\lceil x \rceil$ is the smallest integer larger than x. No further alignment is used. In case of decoding failure of an RS codeword, the GOBs (group of blocks) covered by the RS code will be simply discarded and followed by error concealment. If a GOB is corrupted, the decoder simply drop the GOB and performs a simple error concealment as follows: the motion vector (MV) of a corrupted MB is replaced by the MV of the MB in the GOB above. If the GOB above is also lost, the MV is set to zero, then the MB is replaced by the corresponding MB at the same location in the previous frame. To facilitate error concealment at the decoder when errors occur, the GOBs which are indexed by even numbers are concatenated together, followed by concatenated GOBs indexed by odd numbers. By using this alternative GOB organization, the neighboring GOBs are normally not protected within the same RS codeword. Thus, when a decoding failure occurs in one RS codeword, the neighboring GOBs will not be corrupted simultaneously, which helps the decoder to perform error concealment using the neighboring correctly received GOB.

In order to estimate the end-to-end distortion, we need to model the relation between video MB error probability $P_{\mathrm{MB}}(n, k)$ and RS(n, k) decoding failure probability $p_w(n, k)$, that is,

$$P_{\mathrm{MB}}(n, k) \approx \alpha \cdot p_w(n, k). \tag{5}$$

Since no special packetization or alignment is used, one RS codeword may contain part of one GOB/slice or overlap more than one GOB/slice. It is difficult to find the exact relation between $P_{\mathrm{MB}}(n, k)$ and $p_w(n, k)$ because the length of GOB in each frame is varying. Intuitively, α should be between 1 and 2. Experiments are performed to find the suitable α. Figure 2 shows the experiment results of RS codeword failure probability and GOB error probability over Rayleigh fading channels. It turns out that $\alpha \approx 1.5$ is a good approximation in average. For a source and channel code pair (r_s, r_c) or RS(n, k), the channel code decoding failure probability $p_w(n, k)$ can be derived from ρ and f_d as described in Sections 2.1 and 2.2, then we have the corresponding video MB error probability $P_{\mathrm{MB}}(n, k)$ from (5). Based on the derived MB error rate $P_{\mathrm{MB}}(n, k)$, a recursive estimation method

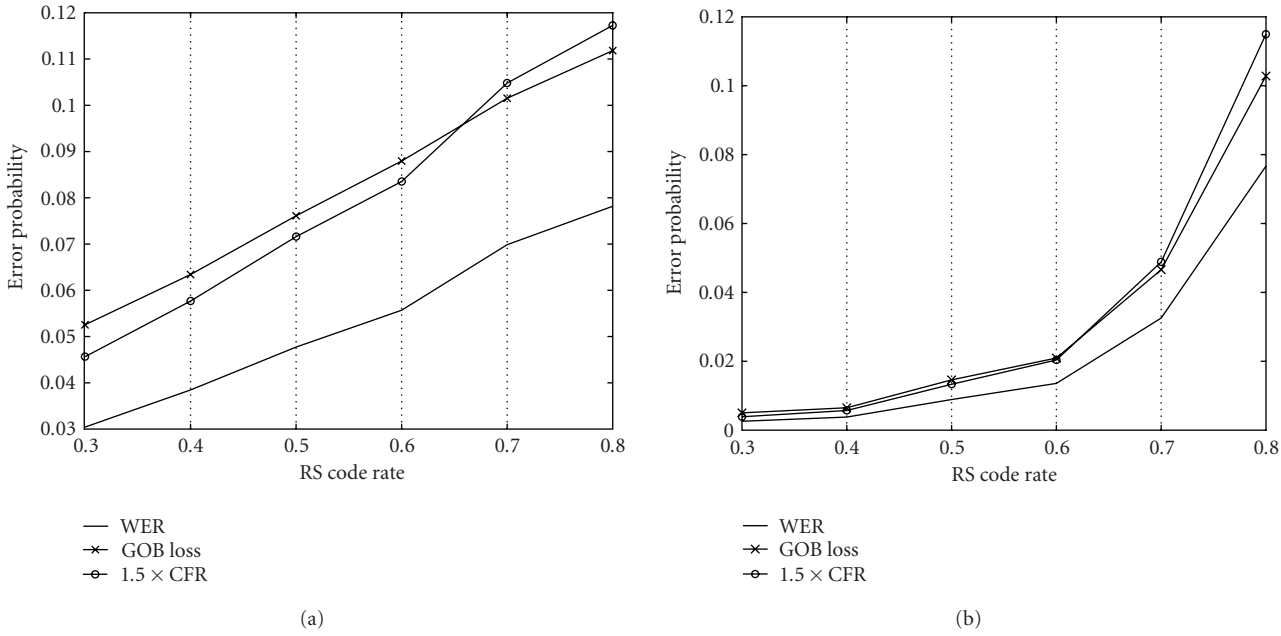


Figure 2: Simulated RS codeword failure rate (CFR), GOB loss rate, and the values of 1.5 + WER. (a) Rayleigh fading, SNR = 18 dB, QPSK, and $fd = 10$ Hz. (b) Rayleigh fading, SNR = 18 dB, QPSK, $fd = 100$ Hz.

and an RD-optimized scheme are employed to estimate the minimal end-to-end MSE of the video sequence and obtain the corresponding optimized video coding scheme, which is to be described in detail in the next section.

3. OPTIMAL DISTORTION ESTIMATION AND MINIMIZATION

We first describe the proposed distortion estimation method for both HP- and IP-based video coding over error-prone channels. Simulations are performed to demonstrate the improved performance of the proposed method. Then an RD framework is used to select the coding mode and quantizer for each MB to minimize the estimated distortion, given the source rate r_s, P_{MB} which is derived as in Section 2, and the frame rate F.

3.1. Optimal distortion estimation

Recently, modeling of error propagation effects have been considered in order to optimally select the mode for each MB to trade off the compression efficiency and error robustness [11, 13, 19]. In particular, a recursive optimal per-pixel estimate (ROPE) of decoder distortion was proposed in [13] which can model the error propagation and quantization distortion more accurately than other methods. But the method in [13] is only optimal for the IP-based video coding. For the HP case, the computation of spatial cross correlation between pixels in the same and different MBs is needed to obtain the first and second moments of bilinear interpolated HPs, the process is computationally prohibitive. Most of the current video coding use the HP-based method to improve the compression performance. We propose a modified recursive estimate of end-to-end distortion that can take care of both IP- and HP-based video coding.

The expected end-to-end distortion for the pixel $f_l^{\mathbf{s}}$ at $\mathbf{s} = (x, y)$ in frame l is

$$\begin{aligned} d_l^{\mathbf{s}} &= E\left\{\left(f_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}}\right)^2\right\} \\ &= E\left\{\left(f_l^{\mathbf{s}} - \hat{f}_l^{\mathbf{s}} + \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}}\right)^2\right\} \\ &= \left(f_l^{\mathbf{s}} - \hat{f}_l^{\mathbf{s}}\right)^2 + 2\left(f_l^{\mathbf{s}} - \hat{f}_l^{\mathbf{s}}\right)E\left(\hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}}\right) + E\left\{\left(\hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}}\right)^2\right\}, \end{aligned} \tag{6}$$

where $\hat{f}_l^{\mathbf{s}}$ is the quantized pixel value at $\mathbf{s}$ in frame l. Denote $e_l^{\mathbf{s}} = f_l^{\mathbf{s}} - \hat{f}_l^{\mathbf{s}}$ as the quantization error, $\hat{e}_l^{\mathbf{s},\mathbf{v}} = \hat{f}_l^{\mathbf{s}} - \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}}$ as the motion compensation error using MV $\mathbf{v}$, and $\tilde{e}_l^{\mathbf{s}} = \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}}$ as the transmission and error-propagation error. Assuming that $\tilde{e}_l^{\mathbf{s}}$ is an uncorrelated random variable with a zero mean, which is a reasonable assumption when P_{MB} is relatively low as will be shown in the simulations later, we have

$$d_l^{\mathbf{s}} = \left(e_l^{\mathbf{s}}\right)^2 + E\left\{\left(\tilde{e}_l^{\mathbf{s}}\right)^2\right\}. \tag{7}$$

We derive a recursive estimate of $E\{(\tilde{e}_l^{\mathbf{s}})^2\}$ for intra-MB and inter-MB as follows.

Intramode MB

The following three cases are considered.

(1) With the probability $1 - P_{\mathrm{MB}}$, the intra-MB is received correctly and then $\hat{f}_l^{\mathbf{s}} = \tilde{f}_l^{\mathbf{s}}$. As a result, $\tilde{e}_l^{\mathbf{s}} = 0$.

(2) With the probability $(1-P_{\mathrm{MB}})P_{\mathrm{MB}}$, the intra-MB is lost but the MB above is received correctly. Denoting by $\mathbf{v}_c = (x_c, y_c)$ the MV of the MB above, two cases of error concealment are considered depending on whether $\mathbf{v}_c$ is at the HP location or not.

(i) If $\mathbf{v}_c$ is at the IP location, we have $\tilde{f}_l^{\mathbf{s}} = \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c}$. Then after error concealment,

$$\begin{aligned}\tilde{e}_l^{\mathbf{s}} &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}} \\ &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c} \\ &= \hat{f}_l^{\mathbf{s}} - \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c} + \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c} \\ &= \hat{e}_l^{\mathbf{s},\mathbf{v}_c} + \tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c}.\end{aligned} \tag{8}$$

The clipping effect is ignored in the computation.

(ii) If $\mathbf{v}_c$ is at the HP location, without loss of generality, assume that $\mathbf{v}_c = (x_c, y_c)$ is at HP location that is interpolated from four neighbouring IP locations with MVs: $\mathbf{v}_c^1 = (\lfloor x_c \rfloor, \lfloor y_c \rfloor)$, $\mathbf{v}_c^2 = (\lceil x_c \rceil, \lfloor y_c \rfloor)$, $\mathbf{v}_c^3 = (\lfloor x_c \rfloor, \lceil y_c \rceil)$, and $\mathbf{v}_c^4 = (\lceil x_c \rceil, \lceil y_c \rceil)$, where $\lfloor x_c \rfloor$ and $\lceil x_c \rceil$ denote the largest integer that is smaller than x_c and the smallest integer larger than x_c, respectively. We have $\tilde{f}_l^{\mathbf{s}} = (\tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^1} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^2} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^3} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^4})/4$ and

$$\begin{aligned}\tilde{e}_l^{\mathbf{s}} &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}} \\ &= \frac{\hat{f}_l^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^1}}{4} + \frac{\hat{f}_l^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^2}}{4} \\ &\quad + \frac{\hat{f}_l^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^3}}{4} + \frac{\hat{f}_l^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}_c^4}}{4} \\ &= \frac{\hat{e}_l^{\mathbf{s},\mathbf{v}_c^1} + \hat{e}_l^{\mathbf{s},\mathbf{v}_c^2} + \hat{e}_l^{\mathbf{s},\mathbf{v}_c^3} + \hat{e}_l^{\mathbf{s},\mathbf{v}_c^4}}{4} \\ &\quad + \frac{\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^1} + \tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^2} + \tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^3} + \tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^4}}{4}.\end{aligned} \tag{9}$$

(3) With the probability P_{MB}^2, both the current MB and the MB above are lost. The MB in the previous video frame at the same location is repeated, that is, $\mathbf{v}_c = \mathbf{0} = (0,0)$:

$$\begin{aligned}\tilde{e}_l^{\mathbf{s}} &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}} \\ &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}} \\ &= \hat{f}_l^{\mathbf{s}} - \hat{f}_{l-1}^{\mathbf{s}} + \hat{f}_{l-1}^{\mathbf{s}} - \tilde{f}_{l-1}^{\mathbf{s}} \\ &= \hat{e}_l^{\mathbf{s},\mathbf{0}} + \tilde{e}_{l-1}^{\mathbf{s}}.\end{aligned} \tag{10}$$

Combining all of the cases together, we have the following results.

(1) If v_c is at the IP location, then

$$\begin{aligned}E_{\mathrm{Intra}}\left\{(\tilde{e}_l^{\mathbf{s}})^2\right\} &= (1-P_{\mathrm{MB}})P_{\mathrm{MB}}\left\{(\hat{e}_l^{\mathbf{s},\mathbf{v}_c})^2 + E\left[(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c})^2\right]\right\} \\ &\quad + P_{\mathrm{MB}}^2\left\{(\hat{e}_l^{\mathbf{s},\mathbf{0}})^2 + E\left[(\tilde{e}_{l-1}^{\mathbf{s}})^2\right]\right\}.\end{aligned} \tag{11}$$

(2) If v_c is at the HP location in both x and y dimensions, then

$$\begin{aligned}&E_{\mathrm{Intra}}\left\{(\tilde{e}_l^{\mathbf{s}})^2\right\} \\ &= (1-P_{\mathrm{MB}})P_{\mathrm{MB}} \\ &\quad \times\left\{\left(\frac{\hat{e}_l^{\mathbf{s},\mathbf{v}_c^1} + \hat{e}_l^{\mathbf{s},\mathbf{v}_c^2} + \hat{e}_l^{\mathbf{s},\mathbf{v}_c^3} + \hat{e}_l^{\mathbf{s},\mathbf{v}_c^4}}{4}\right)^2\right. \\ &\quad \left. + \frac{E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^1}\right)^2\right] + E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^2}\right)^2\right] + E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^3}\right)^2\right] + E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^4}\right)^2\right]}{16}\right\} \\ &\quad + P_{\mathrm{MB}}^2\left\{(\hat{e}_l^{\mathbf{s},\mathbf{0}})^2 + E\left[(\tilde{e}_{l-1}^{\mathbf{s}})^2\right]\right\}.\end{aligned} \tag{12}$$

The cases when v_c has only x_c or y_c at the HP location can be obtained similarly.

Intermode MB

When an intermode MB is correctly received with the probability $1-P_{\mathrm{MB}}$, the motion compensation error $\hat{e}_l^{\mathbf{s}} = \hat{f}_l^{\mathbf{s}} - \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}}$ and the MV $\mathbf{v}$ are received correctly and are reconstructed from the previous reconstructed frame at the decoder. We again consider two cases depending on whether $\mathbf{v} = (x, y)$ is at the IP or HP location.

(1) If $\mathbf{v}$ is at the IP location, then $\tilde{f}_l^{\mathbf{s}} = \hat{e}_l^{\mathbf{s}} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}}$ and

$$\begin{aligned}\tilde{e}_l^{\mathbf{s}} &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}} = \hat{f}_l^{\mathbf{s}} - (\hat{e}_l^{\mathbf{s}} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}}) \\ &= \underbrace{\hat{f}_l^{\mathbf{s}} - \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}} - \hat{e}_l^{\mathbf{s}}}_{0} + \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}} - \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}} \\ &= \tilde{e}_{l-1}^{\mathbf{s},\mathbf{v}}.\end{aligned} \tag{13}$$

(2) If $\mathbf{v} = (x, y)$ is at the HP location in both x and y dimensions, the prediction is interpolated from four pixels with MVs: $\mathbf{v}^1 = (\lfloor x \rfloor, \lfloor y \rfloor)$, $\mathbf{v}^2 = (\lceil x \rceil, \lfloor y \rfloor)$, $\mathbf{v}^3 = (\lfloor x \rfloor, \lceil y \rceil)$, and $\mathbf{v}^4 = (\lceil x \rceil, \lceil y \rceil)$. Then $\tilde{f}_l^{\mathbf{s}} = \hat{e}_l^{\mathbf{s}} + (\tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^1} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^2} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^3} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^4})/4$. We have

$$\begin{aligned}\tilde{e}_l^{\mathbf{s}} &= \hat{f}_l^{\mathbf{s}} - \tilde{f}_l^{\mathbf{s}} \\ &= \hat{f}_l^{\mathbf{s}} - \left(\hat{e}_l^{\mathbf{s}} + \frac{\tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^1} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^2} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^3} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^4}}{4}\right) \\ &= \left(\frac{\hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}^1} + \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}^2} + \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}^3} + \hat{f}_{l-1}^{\mathbf{s}-\mathbf{v}^4}}{4} + \hat{e}_l^{\mathbf{s}}\right) \\ &\quad - \left(\hat{e}_l^{\mathbf{s}} + \frac{\tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^1} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^2} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^3} + \tilde{f}_{l-1}^{\mathbf{s}-\mathbf{v}^4}}{4}\right) \\ &= \frac{\tilde{e}_{l-1}^{\mathbf{s},\mathbf{v}^1} + \tilde{e}_{l-1}^{\mathbf{s},\mathbf{v}^2} + \tilde{e}_{l-1}^{\mathbf{s},\mathbf{v}^3} + \tilde{e}_{l-1}^{\mathbf{s},\mathbf{v}^4}}{4}.\end{aligned} \tag{14}$$

The results of the other MB loss cases are the same as that of the intra-MB. We have the following two results.

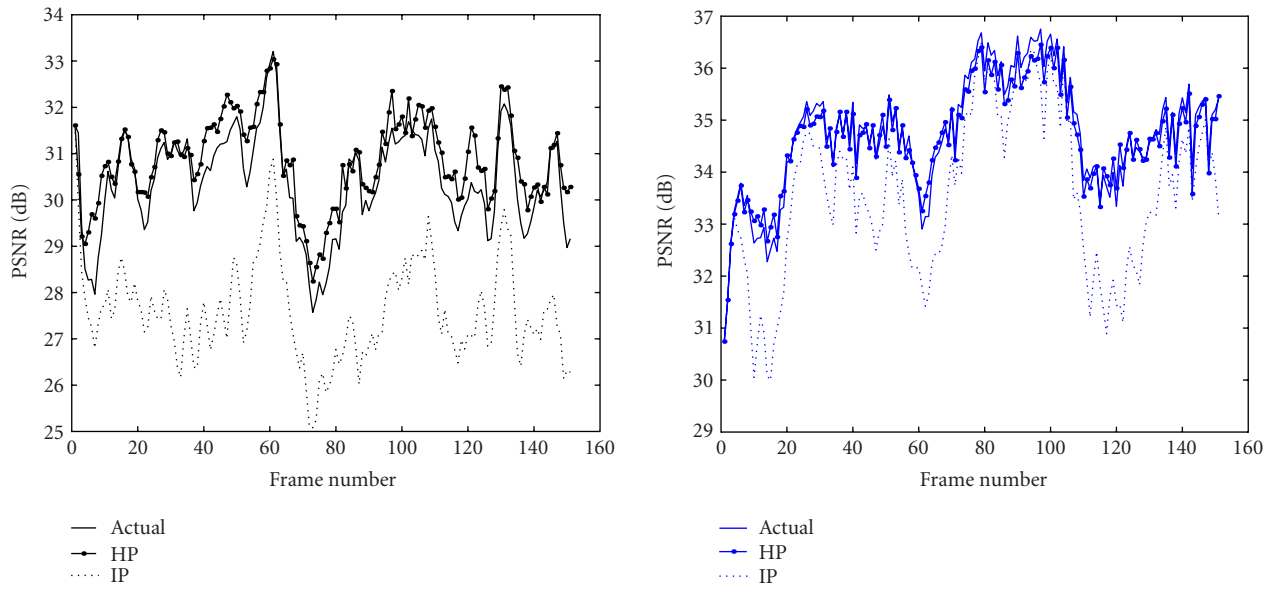


FIGURE 3: Comparison between HP- and IP-based distortion estimation in the HP video coding case (a) Foreman: $r = 300$ kbps, $f = 30$ f/s, and $P_{\mathrm{MB}} = 0.1$. (b) Salesman: $r = 300$ kbps, $f = 30$ f/s, and $P_{\mathrm{MB}} = 0.1$.

(1) If both $\mathbf{v}$ and $\mathbf{v}^c$ are at the IP location, then

$$E_{\mathrm{Inter}}\left\{\left(\tilde{e}_l^{\mathbf{s}}\right)^2\right\} = (1-P_{\mathrm{MB}})E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}}\right)^2\right] + (1-P_{\mathrm{MB}})P_{\mathrm{MB}}\left\{\left(\hat{e}_l^{\mathbf{s},\mathbf{v}_c}\right)^2 + E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c}\right)^2\right]\right\} + P_{\mathrm{MB}}^2\left\{\left(\hat{e}_l^{\mathbf{s},\mathbf{0}}\right)^2 + E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}}\right)^2\right]\right\}. \tag{15}$$

(2) If $\mathbf{v}$ and $\mathbf{v}_c$ are at the HP location in both x and y dimensions, then

$$\begin{aligned} E_{\mathrm{Inter}}&\left\{\left(\tilde{e}_l^{\mathbf{s}}\right)^2\right\} \\ &= (1-P_{\mathrm{MB}}) \\ &\times\left\{\frac{E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}^1}\right)^2\right]+E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}^2}\right)^2\right]+E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}^2}\right)^3\right]+E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}^4}\right)^2\right]}{16}\right\} \\ &+ (1-P_{\mathrm{MB}})P_{\mathrm{MB}} \\ &\times\left\{\left(\frac{\hat{e}_l^{\mathbf{s},\mathbf{v}_c^1}+\hat{e}_l^{\mathbf{s},\mathbf{v}_c^2}+\hat{e}_l^{\mathbf{s},\mathbf{v}_c^3}+\hat{e}_l^{\mathbf{s},\mathbf{v}_c^4}}{4}\right)^2\right. \\ &\quad\left.+\frac{E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^1}\right)^2\right]+E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^2}\right)^2\right]+E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^3}\right)^2\right]+E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}-\mathbf{v}_c^4}\right)^2\right]}{16}\right\} \\ &+ P_{\mathrm{MB}}^2\left\{\left(\hat{e}_l^{\mathbf{s},\mathbf{0}}\right)^2 + E\left[\left(\tilde{e}_{l-1}^{\mathbf{s}}\right)^2\right]\right\}. \end{aligned} \tag{16}$$

The encoder can use the above procedures to recursively estimate the expected distortion $d_l^{\mathbf{s}}$ in (7), based on the accumulated coding and error propagation effects from the previous video frames and current MB coding modes and quantizers. To implement the HP-based estimation, the encoder needs to store an image for $E\{(\tilde{e}_l^{\mathbf{s}})^2\}$; for the locations in which either x or y is at HP, the value is obtained by scaling the sum of the neighboring two values by 1/4, and for locations in which both x and y are at HP precision, it is obtained by scaling the sum of the neighboring four values by 1/16. It should be noted that the scaling by 1/4 or 1/16 can be done by simple bitshift. Both IP- and HP-based estimations need the same memory size to store either two IP images, $E\{f_l\}$ and $E\{f_l^2\}$, or one HP image, $E\{(\tilde{e}_l^{\mathbf{s}})^2\}$, but $E\{(\tilde{e}_l^{\mathbf{s}})^2\}$ requires smaller bitwidth/pel since it is an error signal instead of a pixel value. The HP-based computational complexity is less than the IP-based method since it only needs to compute $E\{(\tilde{e}_l^{\mathbf{s}})^2\}$ instead of computing both $E\{f_l\}$ and $E\{f_l^2\}$ in the IP-based estimate.

We now compare the accuracy of the proposed HP-based estimation to the original IP-based method (ROPE) in [13]. In the simulation, each GOB is carried by one packet. So the packet loss rate is equivalent to the MB error probability P_{MB}. A memoryless packet loss generator is used to drop the packet at a specified loss probability. QCIF sequences *Foreman* and *Salesman* are encoded by the Telenor H.263 encoder with the intra-MB fresh rate set to 4, that is, each MB is forced to be intramode coded if it has not been intracoded for consecutive four frames. The HP- and IP-based estimates are compared to the actual decoder distortion averaged over 50 different channel realizations.

In Figure 3a, the sequence Foreman of 150 frames is encoded with HP motion compensation at a bit rate of

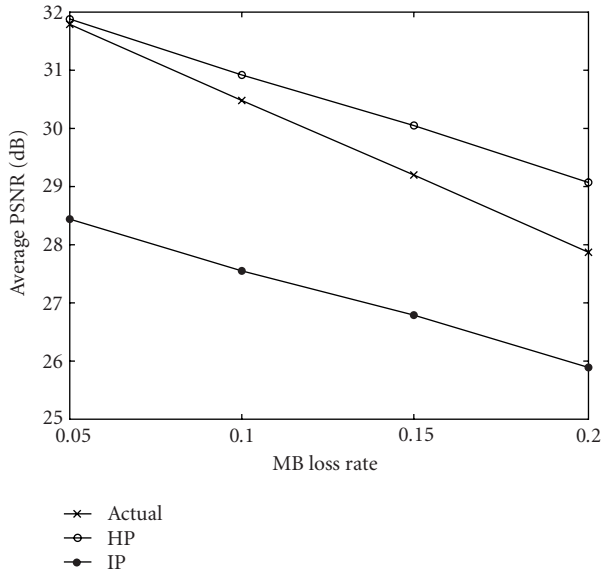


Figure 4: Average PSNR versus MB loss rate for HP- and IP-based distortion estimation; Foreman: $r = 300$ kb/s and $f = 30$ f/s.

300 Kbps, frame rate of 30 f/s, and MB loss rate of 10%. In Figure 3b, the sequence Salesman is encoded in the same way. It can be noted that the HP-based estimation is more accurate to estimate the actual distortion at the decoder compared to the IP-based estimation. Figure 4 also shows the average PSNR of the 150 coded frames with respect to MB loss rates from 5% to 20%. When MB loss rate is as small as 5%, the HP-based estimation is almost the same as the actual distortion, while the IP-based method has about 3 dB difference. The results is as expected since there is about 2–4 dB PSNR difference between HP- and IP-based video coding efficiency given the same bit rate. As the MB loss rate increases as large as 20%, the HP-based estimation is about 1 dB better than the actual distortion, while the IP-based estimation is about 2 dB worse. So the HP-based method is still 1 dB more accurate than the IP-based method. The reason is that the error propagation effects play a more significant role when MB loss rate gets larger, so the coding gain of the HP-based motion compensation is reduced. Also, the assumption in HP-based method that the transmission and propagation errors are not correlated and zero mean may become loose. For practical scenarios, it is demonstrated that the HP-based estimation outperforms the original IP-based method by about 1–3 dB.

3.2. Rate-distortion-optimized video coding

The quantizer step size and code mode for each MB in a frame is optimized by an RD framework. Denote by $b_{i,j,l}$ the MB at location (i, j) in frame l, where $i \in \{1, 2, \ldots, H\}$ and $j \in \{1, 2, \ldots, V\}$. Let $q_{i,j,l} \in Q$, $Q = \{1, 2, \ldots, 32\}$, be the quantizer parameter for $b_{i,j,l}$, and let $m_{i,j,l} \in M$ be the encoding mode for $b_{i,j,l}$, where $M = \{\text{intra}, \text{inter}, \text{skip}\}$ is the set of all admissible encoding modes. Denote by $c_{i,j,l} = [q_{i,j,l}, m_{i,j,l}] \in C$ the encoding vector for $b_{i,j,l}$, where $C = Q \times M$ is the set of all admissible encoding vectors. For each source/channel pair (r_s, r_c), we have the corresponding $P_{\text{MB}}(n, k)$ from (5). The encoder needs to determine the coding mode and quantizer for each MB in total L frames to minimize the end-to-end MSE $D_E(r_s, P_{\text{MB}})$ of the video sequence, which is defined as

$$D_E(r_s, P_{\text{MB}}) = \sum_{l=1}^{L} D_l(R_l, P_{\text{MB}}), \tag{17}$$

where R_l is the number of bits used to encode frame l, its maximal value is denoted as $R_l^{\max} = r_s/F + \Delta_l$ which is the maximal number of bits available to encode frame l provided by a frame level rate control algorithm with average r_s/F and buffer related variable Δ_l. Moreover $D_l(R_l, P_{\text{MB}})$ is the estimated end-to-end MSE of frame l, $l = 1, 2, \ldots, L$, which can be obtained as

$$D_l(R_l, P_{\text{MB}}) = \sum_{i=1}^{H} \sum_{j=1}^{V} D(c_{i,j,l}, P_{\text{MB}}), \tag{18}$$

where $D(c_{i,j,l}, P_{\text{MB}})$ is the end-to-end MSE of MB $b_{i,j,l}$ using encoding vector $c_{i,j,l}$ and $D(c_{i,j,l}, P_{\text{MB}})$ can be computed from $d_l^{\mathbf{s}}$ as

$$D(c_{i,j,l}, P_{\text{MB}}) = \sum_{\mathbf{s} \in b_{i,j,l}} d_l^{\mathbf{s}}. \tag{19}$$

Since there is dependency between neighboring interframes because of the motion compensation, the optimal solution of (17) has to be searched over $C^{H \times V \times L}$, which is computationally prohibitive. We use greedy optimization algorithm, which is also implicitly used in most JSCC video coding methods such as [10, 11, 13], to find the coding modes and quantizers for MBs in frame l that minimize $D_l(R_l, P_{\text{MB}})$, then find coding modes and quantizers for MBs in frame $l+1$ that minimize $D_{l+1}(R_{l+1}, P_{\text{MB}})$ based on the previous optimized frame l, and so on. The optimal pair (r_s^*, r_c^*) and the corresponding optimal video coding scheme can be found such that

$$(r_s^*, r_c^*) = \underset{(r_s, r_c)}{\operatorname{argmin}} D_E^*(r_s, P_{\text{MB}}). \tag{20}$$

The goal now is to optimally select the quantizers and encoding modes on the MB level for a specific MB error rate P_{MB} and frame rate $R_l^{\max}$ to trade off the source coding efficiency and robustness to error. The notation of P_{MB} and (r_s, r_c) is dropped from now on unless needed.

The optimal coding problem for frame l can be stated as

$$\min_{C^{H \times V}} D_l = \sum_{i=1}^{H} \sum_{j=1}^{V} D(c_{l,i,j}) \tag{21}$$

subject to

$$R_l = \sum_{i=1}^{H} \sum_{j=1}^{V} R(c_{l,i,j}) \leq R_l^{\max}. \quad (22)$$

Such RD-optimized video coding schemes have been studied for noiseless and noisy channels recently [19, 20, 21, 22, 23, 24]. Using Lagrangian multiplier, we can solve the problem by minimizing

$$J_l(\lambda) = D_l + \lambda R_l, \quad (23)$$

where $\lambda \geq 0$. For video coding over error-prone channels, GOB coding structure is used for H.263 video coding over noisy channels with each GOB encoded independently. Therefore, if the transmission errors occur in one GOB, the errors will not propagate into other GOBs in the same video frame.

For video coding over noiseless channels, the independent GOB structure leads to the fact that the optimization of (23) can be performed for each GOB separately. However, when considering RD-optimized video coding for noisy channels, the MB distortion $D_{i,j}(c_{i,j}, P_{\mathrm{MB}})$ depends not only on the mode and quantizer of the current MB but also on the mode of the MB above to take into account error concealment distortion. Therefore, there is a dependency between neighboring GOBs for this optimization problem. We use greedy optimization algorithm again to find the solution by searching the optimal modes and quantizers from the first GOB to the last GOB in each frame.

4. SIMULATION RESULTS

We first use a simple two-state Markov chain model for simulation to show the performance of the integrated source and channel rate allocation and robust video coding scheme, where the given channel stochastic knowledge is accurate. Then simulations over Rayleigh fading channel is performed to verify the effectiveness of the proposed scheme for practical wireless channels.

4.1. Two-state Markov chain channel

Simulations have been performed using base mode H.263 to verify the accuracy of the proposed integrated scheme. In the simulations, the total channel signaling rate r equals 144 kbps, which is a typical rate provided in the 3G wireless systems. Video frame rate is $F = 10$ f/s. The video sequence used for simulation is Foreman in QCIF format. RS code over $GF(2^8)$ is used for FEC. The channel coding rate used are $\{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. The source and channel coding rates r_s, r_c and the corresponding RS code (n, k) are listed in Table 1. A two-state Markov channel model [16] is used, where the state transition is at the RS symbol level. The two states of the model are denoted by G (good) and B (bad). In state G, the symbols are received correctly ($e_g = 0$) whereas in state B, the symbols are erroneous ($e_b = 1$). The model is fully described by the transition probabilities p from state G to state B, and q from state B to state G. We use the probability of state B:

$$P_B = \frac{p}{p+q}, \quad (24)$$

and the average bursty length:

$$L_B = \frac{1}{q}, \quad (25)$$

which is the average number of consecutive symbol errors to model the two-state Markov model [11, 16].

The simulations are performed through the following steps.

(i) For each channel coding rate r_c (or RS(n, k)) in each column of Table 1, the RS code decoding failure rate $p_w(n, k)$ is computed using (3) for a given two-state Markov channel model. The results for different r_c and channel models are shown in Table 2.

(ii) The corresponding video MB error rate $P_{\mathrm{MB}}(r_c)$ is obtained using (5), where $\alpha = 1.5$.

(iii) For each source rate $r_s = r \times r_c$ and the corresponding $P_{\mathrm{MB}}(r_c)$, the RD-optimized H.263 video coding is employed while estimating the end-to-end MSE $D_E^*(r_s, r_c)$.

(iv) The H.263 bitstream is packetized and protected using RS(n, k), and then transmitted over the two-state Markov channel model.

(v) The receiver receives the bitstream, reconstructs the video sequence after the FEC decoding, and performs the H.263 decoding and possible error concealment if errors occur. The distortion for each simulation run between the original video sequence and the reconstructed video sequence at the receiver is also computed.

The average estimated PSNR, PSNR_E, of video signals is used to measure the performance:

$$\mathrm{PSNR}_E(r_s, r_c) = \frac{1}{L} \sum_{l=1}^{L} \mathrm{PSNR}_E^l(r_s, r_c), \quad (26)$$

where

$$\mathrm{PSNR}_E^l(r_s, r_c) = 10 \log_{10} \frac{255^2}{D_E^*(r_s, r_c)} \quad (27)$$

is the estimated average PSNR between the original frame l and the corresponding reconstruction at the decoder using the pair (r_s, r_c), and $D_E^*(r_s, r_c)$ is the minimal estimated end-to-end MSE from (17) through RD-optimized video coding. The average PSNR of N runs of simulation is defined as

$$\mathrm{PSNR}_S(r_s, r_c) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{L} \sum_{l=1}^{L} \mathrm{PSNR}_S^{(n,l)}(r_s, r_c), \quad (28)$$

where $\mathrm{PSNR}_S^{(n,l)}(r_s, r_c)$ is the PSNR between the original

Table 1: The source and channel rates used in simulation.

r (kbps)	144	144	144	144	144	144	144
RS(n,k)	$(200,40)$	$(200,60)$	$(200,80)$	$(200,100)$	$(200,120)$	$(200,140)$	$(200,160)$
$r_c = k/n$	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$r_s = r \times r_c$ (kbps)	28.8	43.2	57.6	72.0	86.4	100.8	115.2

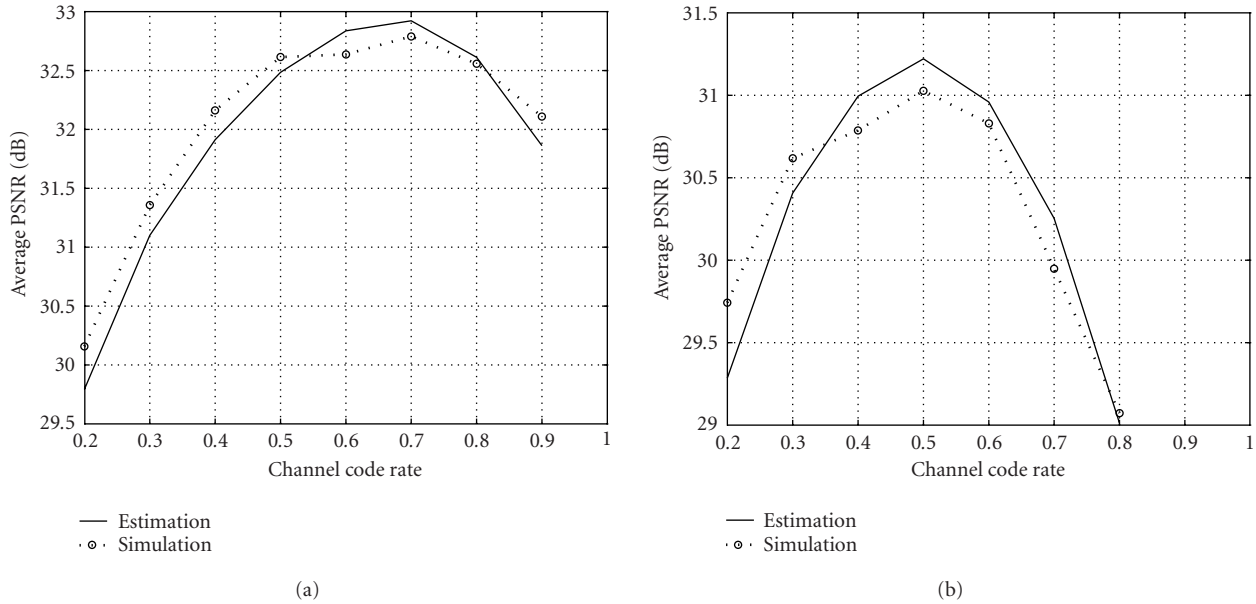


Figure 5: Average PSNR obtained by estimation versus simulation using two-state Markov model. (a) Symbol error rate = 0.01, where $P_B = 0.01$ and $L_B = 16$. (b) Symbol error rate = 0.05, where $P_B = 0.05$ and $L_B = 16$.

frame l and the corresponding reconstruction at the decoder in the nth simulation using the source/channel rate pair (r_s, r_c).

Figure 5a shows the average estimated PSNR_E of the optimal rate allocation and robust video coding for different channel code rates when the symbol error rate is $P_B = 0.01$ and the bursty length $L_B = 16$ symbols, and the corresponding average simulated PSNR_S is of 50 times video transmission. Figure 5b also shows the same comparison when the symbol error rate is $P_B = 0.05$. It can be noted that the estimated PSNR_E, which is obtained at the encoder during RD-optimized video encoding, matches the simulated PSNR_S very well. The optimal source and channel rate pair can also be found through Figures 5a and 5b for different channel characteristics. The corresponding channel decoding failure rate of the optimal channel coding rates in Figures 5a and 5b are 0.018 and 0.034, respectively.

We also compare the performance when the knowledge of channel model used at video encoder does not match the real channel used in simulations. Figure 6 shows two cases of channel mismatch. In Figure 6a, the video stream, which is encoded based on $P_B = 0.01$ and $L_B = 16$ two-state Markov channel, is simulated using two-state Markov channel with $P_B = 0.01$ and $L_B = 8$. The simulated average PSNR is better than the average PSNR estimated at the encoder during encoding because the channel model used in estimation is worse than the model used in simulation. On the other hand, when the video stream, which is encoded based on $P_B = 0.01$ and $L_B = 8$ two-state Markov channel, is simulated using two-state Markov channel with $P_B = 0.01$ and $L_B = 16$, the simulated average PSNR is much worse than the average PSNR estimated at the encoder as shown in Figure 6b. Furthermore, the optimal source and channel coder pair obtained at the encoder is not optimal when the channel condition used in simulation is worse than the channel information used at the encoder. This simulation result suggests that the optimal rate allocation and video coding should be focused on the worse channel conditions for broadcasting services.

4.2. Rayleigh fading channel

The simulation over the Rayleigh fading channel is also performed to verify the effectiveness of the proposed scheme over realistic wireless channels. In the simulation, QPSK with coherent demodulation is used for the sake of simplicity. The channel is a frequency-nonselective Rayleigh fading channel. An FSMC with $K = 6$ states is used to model the Rayleigh fading channel. The SNR thresholds for the K states are

Table 2: The Reed-Solomon code decoding failure rate.

Rate	$P_B = 0.01, L_B = 16$	$P_B = 0.05, L_B = 16$	$P_B = 0.01, L_B = 8$	$P_B = 0.05, L_B = 8$
0.1	0.00028	0.00266	0.00000	0.00006
0.2	0.00058	0.00521	0.00001	0.00023
0.3	0.00117	0.00998	0.00003	0.00079
0.4	0.00233	0.01871	0.00011	0.00259
0.5	0.00462	0.03435	0.00042	0.00803
0.6	0.00909	0.06170	0.00155	0.02342
0.7	0.01776	0.10840	0.00559	0.06384
0.8	0.03445	0.18603	0.01976	0.16098
0.9	0.06635	0.31135	0.06829	0.36890

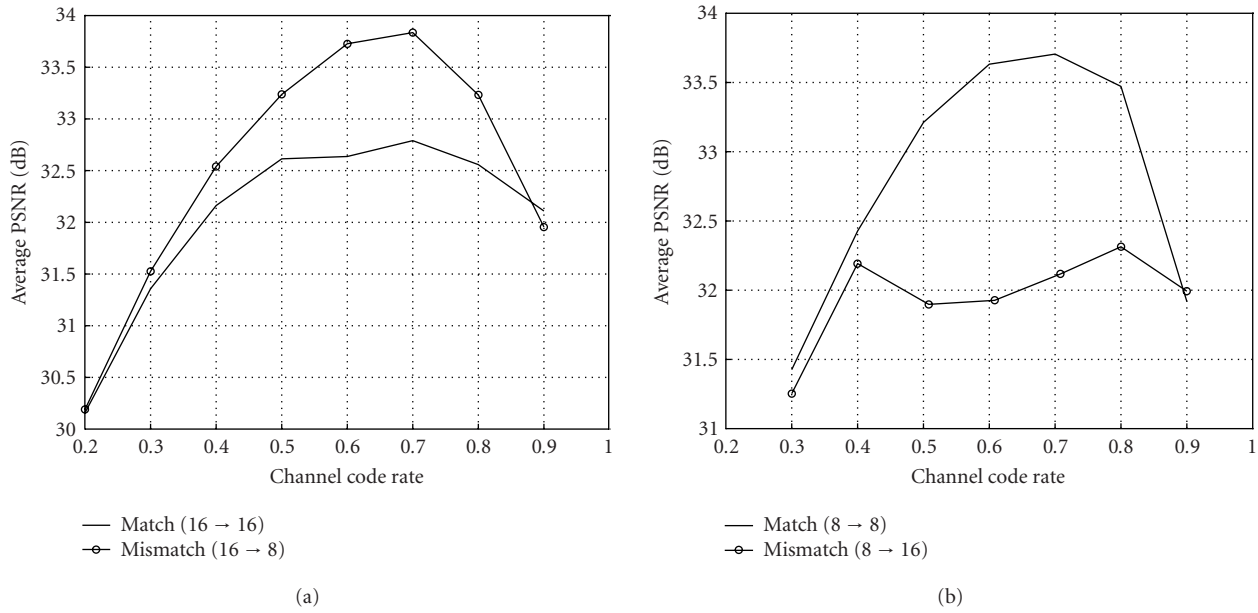


Figure 6: Average PSNR obtained in channel mismatch cases. (a) Error burst is shorter than that used in estimation. (b) Error burst is longer than that used in estimation.

selected in such a way that the probability that the channel gain is at state s_k, $k = 0, 1, \ldots, K-1$, is

$$p_0 = \frac{2}{K(K+1)}, \qquad p_k = kp_0, \quad k = 1, 2, \ldots, K-1. \tag{29}$$

The FSMC state transition is described at the RS codeword symbol level (8-bit RS symbol) with the assumption that the four QPSK modulation symbols within an RS codeword symbol stay in the same FSMC state. Given the average SNR ρ and the Doppler frequency f_d, we can obtain the parameters such as steady state probability p_k, RS symbol error probability e_k, and state transition rates [17]. Then following the procedures described in Section 2.1, we are able to analyze the RS code performance over Rayleigh fading channels. Table 3 shows the estimated RS code decoding failure probability using FSMC model and the simulation values when the SNR is 18 dB and the Doppler frequency is 10 Hz and 100 Hz, respectively. The RS codeword error rate obtained by the FSMC matches the simulation results very well when f_d is 10 Hz. When f_d is 100 Hz, the FSMC-based estimate is not as accurate as the results when f_d is 10 Hz, but is still within acceptable range compared to the simulated values.

Figure 7a shows the average estimated PSNR_E and simulated PSNR_S of the video coding after optimal rate allocation and robust video coding for different channel code rates when the SNR is 18 dB and f_d is 10 Hz. Figure 7b also shows the comparison when the fd is 100 Hz. Even though it can be noted that there are about 1 dB difference between the estimated PSNR_E and the simulated PSNR_S, the near-optimal source and channel rate allocation (or the channel code rate

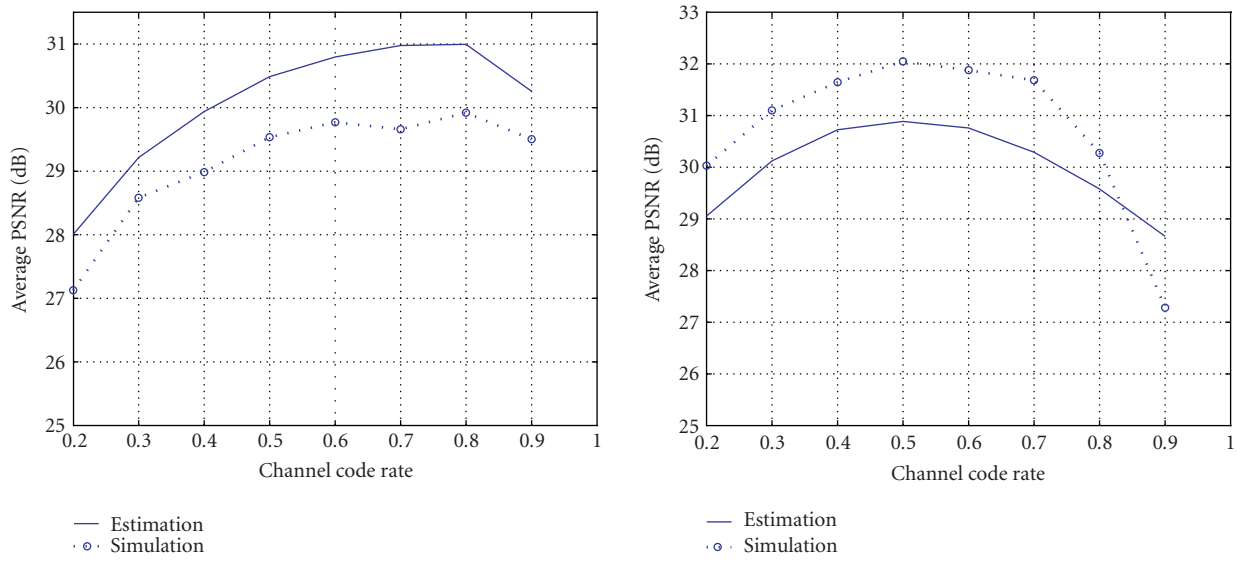


Figure 7: Average end-to-end PSNR obtained by estimation versus simulation for Rayleigh fading channels: (a) SNR = 18 dB, f_d = 10 Hz, and (b) SNR = 18 dB, f_d = 100 Hz.

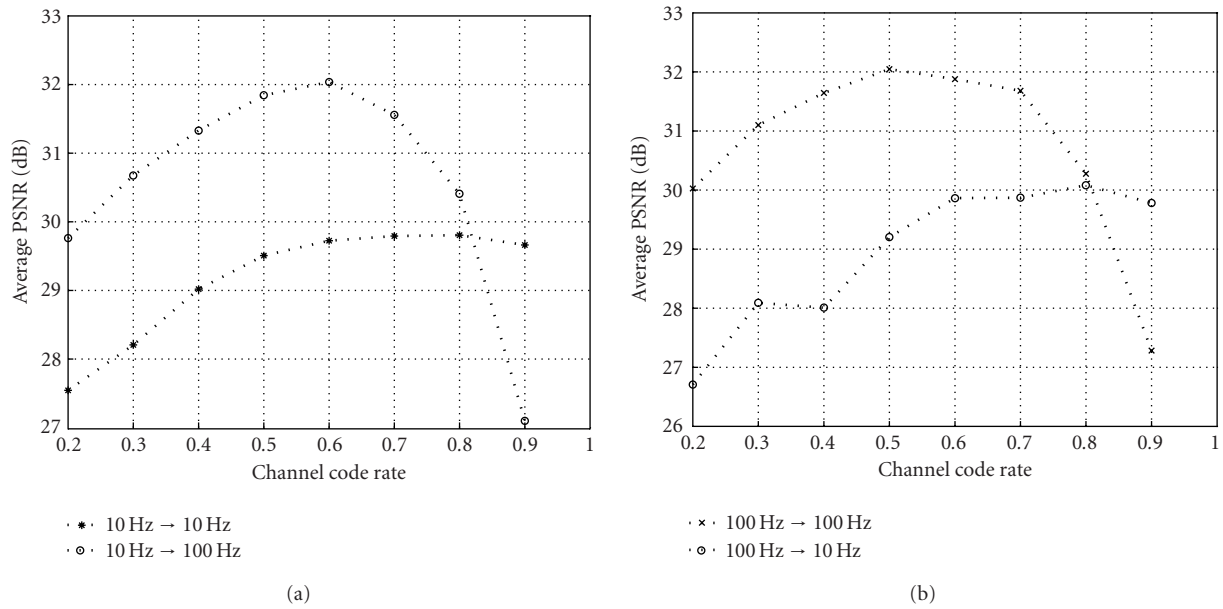


Figure 8: Average end-to-end PSNR over Rayleigh fading Channels: (a) SNR = 18 dB, f_d used for estimation at the encoder is 10 Hz, and (b) SNR = 18 dB, f_d used for estimation at the encoder is 100 Hz.

r_c) obtained from the estimation (0.8 and 0.5 as shown in Figure 7) still has the maximal simulated end-to-end PSNR over Rayleigh fading channels. The simulation results verify the effectiveness of the proposed scheme to obtain the optimal source and channel coding pair when given a fixed total bit rate for wireless fading channels.

Experiments are also performed when the knowledge of channel Doppler frequency used at the video encoder does not match the actual Doppler frequency used in simulations. Figure 8 shows two cases of channel mismatch. In Figure 8a, the video bitstream which is encoded based on f_d = 10 Hz is simulated over fading channels with Doppler frequency

Table 3: Analysis and simulation values of the RS code decoding failure probability for the Rayleigh fading channel with SNR equals 18 dB and the Doppler frequency f_d equals 10 and 100 Hz.

	$f_d = 10$ Hz		$f_d = 100$ Hz	
Code rate	FSMC model	Simulation	FSMC model	Simulation
0.2	0.0430	0.0391	0.0098	0.0044
0.3	0.0482	0.0464	0.0170	0.0072
0.4	0.0536	0.0555	0.0282	0.0119
0.5	0.0593	0.0650	0.0450	0.0181
0.6	0.0653	0.0772	0.0692	0.0280
0.7	0.0717	0.0915	0.1032	0.0526
0.8	0.0815	0.1085	0.1499	0.1110
0.9	0.1240	0.1333	0.2131	0.2856

of 10 Hz and 100 Hz, separately. In Figure 8b, the video bitstream which is encoded based on $f_d = 100$ Hz is simulated over fading channels with Doppler frequency of 100 Hz and 10 Hz, separately. In both scenarios, the video quality would be better if the actual condition in terms of MB loss rate is smaller than the knowledge used at the encoder, and would be worse otherwise. Furthermore, the optimal source and channel coder pair obtained at the encoder is not optimal when the channel condition used in simulation is worse than the channel information used at the encoder. This simulation result again suggests that the optimal rate allocation and video coding should be focused on the worse channel conditions for broadcasting services.

5. CONCLUSION

We have proposed an integrated framework to find the near-optimal source and channel rate allocation and the corresponding robust video coding scheme for video coding and transmission over wireless channels when there is no feedback channel available. Assuming that the encoder has the stochastic channel information when the wireless fading channel is modeled as an FSMC model, the proposed scheme takes into account the robust video coding, packetization, channel coding, error concealment, and error propagation effects altogether. This scheme can select the best source and channel coding pair to encode and transmit the video signals. Simulation results demonstrated the optimality of the rate allocation scheme and accuracy of end-to-end MSE estimation obtained at the encoder during the process of robust video encoding.

REFERENCES

[1] B. Girod and N. Farber, "Feedback-based error control for mobile video transmission," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1707–1723, 1999.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Series in Communications. John Wiley & Sons, NY, USA, 1991.

[3] J. Modestino and D. G. Daut, "Combined source-channel coding of images," *IEEE Trans. Communications*, vol. 27, pp. 1644–1659, November 1979.

[4] N. Tanabe and N. Farvardin, "Subband image coding using entropy-coded quantization over noisy channels," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 5, pp. 926–943, 1992.

[5] N. Farvardin and V. Vaishampayan, "On the performance and complexity of channel-optimized vector quantizers," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 155–160, 1991.

[6] G. Cheung and A. Zakhor, "Bit allocation for joint source/channel coding of scalable video," *IEEE Trans. Image Processing*, vol. 9, no. 3, pp. 340–356, 2000.

[7] M. Bystrom and J. W. Modestino, "Combined source-channel coding schemes for video transmission over an additive white Gaussian noise channel," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 880–890, 2000.

[8] 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, no. 10, pp. 1734–1750, 1999.

[9] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, 2000.

[10] M. Gallant and F. Kossentini, "Rate-distortion optimized layered coding with unequal error protection for robust internet video," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 357–372, 2001.

[11] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1012–1032, 2000.

[12] Z. He, J. Cai, and C. W. Chen, "Joint source channel rate-distortion analysis for adaptive mode selection and rate control in wireless video coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 511–523, 2002.

[13] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra mode switching for packet loss resilience," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 966–976, 2000.

[14] ITU-T, "H.263, video coding for low bitrate communication," February 1998.

[15] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[16] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell System Technical Journal*, vol. 42, pp. 1977–1997, September 1963.

[17] H. Wang and N. Moayeri, "Finite-state Markov channel—a usefulmodel for radio communication channels," *IEEE Trans. Vehicular Technology*, vol. 44, no. 1, pp. 163–171, 1995.

[18] J. Lu, K. B. Letaief, and M. L. Liou, "Robust video transmission over correlated mobile fading channels," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 5, pp. 737–751, 1999.

[19] G. Cote, S. Shirani, and F. Kossentini, "Robust H.263 video communication over mobile channels," in *IEEE Proc. of International Conference on Image Processing*, pp. 535–539, Kobe, Japan, October 1999.

[20] G. M. Schuster and A. K. Katsaggelos, "Fast and efficient mode and quantizer selection in the rate distortion sense for H.263," in *Proc. SPIE: Visual Communications and Image Processing*, vol. 2727, pp. 784–795, Orlando, USA, February 1996.

[21] H. Sun, W. Kwok, M. Chien, and C. H. J. John, "MPEG coding performance improvement by jointly optimizing coding mode decisions and rate control," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 449–458, 1997.

[22] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, 1998.

[23] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[24] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. Mitra, "Rate-distortion optimized mode selection for very low bit-rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 182–190, 1996.

Jie Song received his B.S. degree from Beijing University, Beijing, China, in 1990, his M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1993, and his Ph.D. degree from the University of Maryland, College Park, Md, in 2000, all in electrical engineering. From April 1993 to June 1996, he was a Lecturer and a Researcher in the Information Engineering Department at Beijing University of Posts and Telecommunications. He worked for Fujitsu Labs of America, Calif, in the summer of 1997. From November 1997 to February 1999, he was a part-time Consultant of multimedia technologies in Odyssey Technologies Inc., Jessup, Md, where he was involved in the projects of H.323/H.324 videophone, portable multimedia terminal design, and multichannel video capturing systems. Since August 2000, he has been working on research, design, and implementation for broadband and satellite communication systems at Agere Systems (formerly Microelectronics Group, Lucent Technologies). His research interests include signal processing for digital communication and multimedia communications.

K. J. Ray Liu received the B.S. degree from the National Taiwan University in 1983, and the Ph.D. degree from UCLA in 1990, both in electrical engineering. He is a Professor at the Electrical and Computer Engineering Department and Institute for Systems Research of the University of Maryland, College Park. His research contributions encompass broad aspects of signal processing algorithms and architectures; multimedia communications and signal processing; wireless communications and networking; information security; and bioinformatics, in which he has published over 300 refereed papers. Dr. Liu is the recipient of numerous honors and awards including IEEE Signal Processing Society 2004 Distinguished Lecturer, the 1994 National Science Foundation Young Investigator Award, the IEEE Signal Processing Society's 1993 Senior Award (Best Paper Award), IEEE 50th Vehicular Technology Conference Best Paper Award, Amsterdam, 1999. He also received the George Corcoran Award in 1994 for outstanding contributions to electrical engineering education and the Outstanding Systems Engineering Faculty Award in 1996 in recognition of outstanding contributions in interdisciplinary research, both from the University of Maryland. Dr. Liu is a Fellow of IEEE. Dr. Liu is the Editor-in-Chief of IEEE Signal Processing Magazine and was the founding Editor-in-Chief of EURASIP Journal on Applied Signal Processing. Dr. Liu is a Board of Governor and has served as Chairman of Multimedia Signal Processing Technical Committee of IEEE Signal Processing Society.

EURASIP Journal on Applied Signal Processing 2004:2, 317–329

Medusa: A Novel Stream-Scheduling Scheme for Parallel Video Servers

Hai Jin
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
Email: hjin@hust.edu.cn

Dafu Deng
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
Email: dfdeng@hust.edu.cn

Liping Pang
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
Email: lppang@hust.edu.cn

Received 6 December 2002; Revised 15 July 2003

Parallel video servers provide highly scalable video-on-demand service for a huge number of clients. The conventional stream-scheduling scheme does not use I/O and network bandwidth efficiently. Some other schemes, such as batching and stream merging, can effectively improve server I/O and network bandwidth efficiency. However, the batching scheme results in long startup latency and high reneging probability. The traditional stream-merging scheme does not work well at high client-request rates due to mass retransmission of the same video data. In this paper, a novel stream-scheduling scheme, called *Medusa*, is developed for minimizing server bandwidth requirements over a wide range of client-request rates. Furthermore, the startup latency raised by Medusa scheme is far less than that of the batching scheme.

Keywords and phrases: video-on-demand, stream batching, stream merging, multicast, unicast.

1. INTRODUCTION

In recent years, many cities around the world already have, or are deploying, the *fibre to the building* (FTTB) network on which users access the optical fibre *metropolitan area network* (MAN) via the fast LAN in the building. This kind of large-scale network improves the end bandwidth up to 100 Mb per second and has enabled the increasing use of larger-scale *video-on-demand* (VOD) systems. Due to the high scalability, the parallel video servers are often used as the service providers in those VOD systems.

Figure 1 shows a diagram of the large-scale VOD system. On the client side, users request video objects via their PCs or dedicated set-top boxes connected with the fast LAN in the building. Considering that the 100 Mb/s Ethernet LAN is widely used as the in-building network due to its excellent cost/effective rate, we only focus on the clients with such bandwidth capacity and consider the VOD systems with homogenous client network architecture in this paper.

On the server side, the parallel video servers [1, 2, 3] have two logical layers. Layer 1 is an RTSP server, which is responsible for exchanging the RTSP message with clients and scheduling different RTP servers to transport video data to clients. Layer 2 consists of several RTP servers that are responsible for concurrently transmitting video data according to the RTP/RTCP. In addition, video objects are often striped into lots of small segments that are uniformly distributed among RTP server nodes so that the high scalability of the parallel video servers can be guaranteed [2, 3].

Obviously, the key bottleneck of those large-scale VOD systems is the bandwidth of parallel video servers, either the disk I/O bandwidth of parallel video servers, or the network bandwidth connecting the parallel video servers to the MAN. For using the server bandwidth efficiently, a stream-scheduling scheme plays an important role because it determines how much video data should be retrieved from disks and transported to clients. The conventional scheduling scheme sequentially schedules RTP server nodes to transfer segments of a video object via unicast propagation method. Previous works [4, 5, 6, 7, 8] have shown that most clients often request several hot videos in a short time interval. This makes the conventional scheduling scheme send lots of same

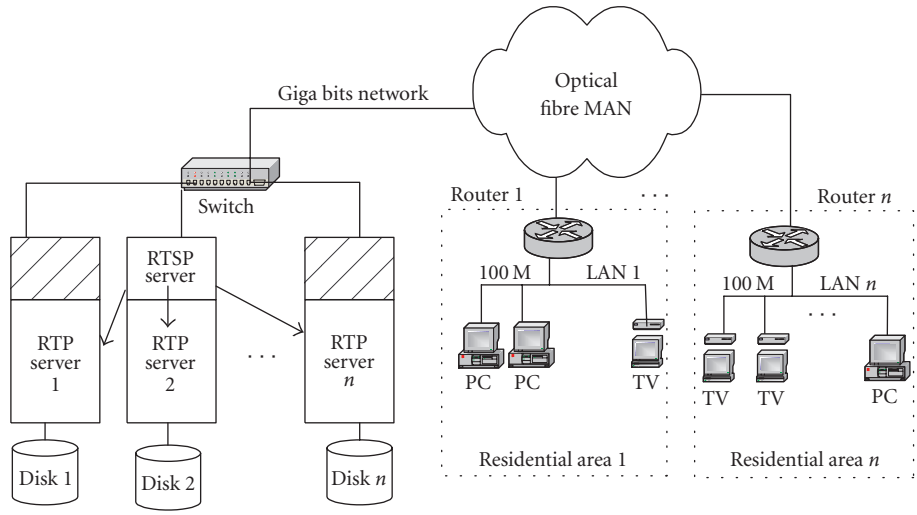


Figure 1: A larger-scale VOD system supported by parallel video servers.

video-data streams during a short time interval. It wastes the server bandwidth and better solutions are necessary.

The multicast or broadcast propagation method presents an attractive solution for the server bandwidth problem because a single multicast or broadcast stream can serve lots of clients that request the same video object during a short time interval. In this paper, we focus on the above VOD system, and then, based on the multicast method, develop a novel stream-scheduling scheme for the parallel video servers, called Medusa, which minimizes the server bandwidth consumption over a wide range of client-request rates.

The following sections are organized as follows. In Section 2, we describe the related works on the above bandwidth efficiency issue and analyze the existing problem of these schemes. Section 3 describes the scheduling rules for the Medusa scheme and Section 4 discusses how to determine the time interval T used in the Medusa scheme. Section 5 presents information of the performance evaluation. Section 6 proposes some discussions for the Medusa scheme. Finally, Section 7 ends with conclusions and future works.

2. RELATED WORKS

In order to use the server bandwidth efficiently, two kinds of schemes based on the multicast or broadcast propagation method have been proposed: the batching scheme and the stream-merging scheme.

The basic idea of the batching scheme is using a single multicast stream of data to serve clients requesting the same video object in the same time interval. Two kinds of batching schemes were proposed: first come first serve (FCFS) and maximum queue length (MQL) [4, 6, 9, 10, 11, 12]. In FCFS, whenever a server schedules a multicast stream, the client with the earliest request arrival is served. In MQL, the incoming requests are put into separate queues based on the requested video object. Whenever a server schedules a multicast stream, the longest queue is served first. In any case, a time threshold must be set first in the batching scheme. Video servers just schedule the multicast stream at the end of each time threshold. In order to obtain efficient bandwidth, the value of this time threshold must be at least 7 minutes [7]. The expected startup latency is approximately 3.5 minutes. The long delay increases the client reneging rate and decreases the popularization of VOD systems.

The stream-merging scheme presents an efficient way to solve the long startup latency problem. There are two kinds of scheduled streams: the complete multicast stream and the patching unicast stream. When the first client request has arrived, the server immediately schedules a complete multicast stream with a normal propagation rate to transmit all of the requested video segments. A later request to the same video object must join the earlier multicast group to receive the remainder of the video, and simultaneously, the video server schedules a new patching unicast stream to transmit the lost video data to each of them. The patching video data is propagated at double video play rate so that two kinds of streams can be merged into an integrated stream.

According to the difference in scheduling the complete multicast stream, stream-merging schemes can be divided into two classes: *client-initiated with prefetching* (CIWP) and *server-initiated with prefetching* (SIWP).

For CIWP [5, 13, 14, 15, 16, 17], the complete multicast stream is scheduled when a client request arrives. The latest complete multicast stream for the same video object cannot be received by that client.

For SIWP [8, 18, 19], a video object is divided into segments, each of which is multicast periodically via a dedicated multicast group. The client prefetches data from one or several multicast groups for playback.

Stream-merging schemes can effectively decrease the required server bandwidth. However, with the increase of client-request rates, the amount of the same retransmitted video data is expanded dramatically and the server

Table 1: Notations and Definitions.

Notations	Definitions
T	The length of time interval and also the length of a video segment (in min)
M	The amount of video objects stored on the parallel video server
N	The amount of RTP server nodes in the parallel video servers
t_i	The ith time interval; the interval in which the first client request arrives is denoted by t_0; the following time intervals are denoted by $t_1, \ldots, t_i, \ldots$, respectively, $(i = 0, \ldots, +\infty)$
L	The length of the requested video object (in min)
$S(i, j)$	The ith segment of the requested object; j represents the serial number of the RTP server node on which this segment is stored
R_i	The client requests arriving in the ith time interval $(i = 0, \ldots, +\infty)$
PS_i	The patching multicast stream initialized at the end of the ith time interval $(i = 0, \ldots, +\infty)$
CS_i	The complete multicast stream initialized at the end of the ith time interval $(i = 0, \ldots, +\infty)$
$\tau(m, n)$	The start transporting time for the mth segment transmitted on the stream PS_n or CS_n
G_i	The client-requests group in which all clients are listening to the complete multicast stream CS_i
b_c	The client bandwidth capacity, in unit of stream (assuming the homogenous client network)
$\mathrm{PB}_{\max}$	The maximum number of patching multicast streams that can be concurrently received by a client
λ_i	The client-request arrival rate for the ith video object

bandwidth efficiency is seriously damaged. Furthermore, a mass of double-rated patching streams may increase the network traffic burst.

3. MEDUSA SCHEME

Because video data cannot be shared among clients requesting for different video objects, the parallel video server handles those clients independently. Hence, we only consider clients requesting for the same hot video object in this section (more general cases will be studied in Section 5).

3.1. The basic idea of the Medusa scheme

Consider that the requested video object is divided into lots of small segments with a constant playback time length T. Based on the value of T, the time line is slotted into fixed-size time intervals and the length of each time interval is T. Usually, the value of T is very small. Therefore, it would not result in long startup latency. The client requests arriving in the same time interval are batched together and served as one request via the multicast propagation method. For convenient description, we regard client requests arriving in the same time interval as one client request in the following sections.

Similar to stream-merging schemes, two kinds of multicast streams, the *complete multicast streams* and the *patching multicast streams*, can be used to reduce the amount of retransmitted video data. A complete multicast stream responses to transporting all segments of the requested video object while a patching multicast stream just transmits partial segments of that video object. The first arrival request is served immediately by a complete multicast stream. Later starters must join the complete multicast group to receive the remainder of the requested video object. At the same time, they must join as more earlier patching multicast groups as possible to receive valid video data. For those really missed video data, the parallel video servers schedule a new patching multicast stream for transporting them to clients.

Note that the IP multicast, the broadcast, and the application-level multicast are often used in VOD systems. In those multicast technologies, a user is allowed to join lots of multicast groups simultaneously. In addition, because all routers in the network would exchange their information periodically, each multicast packet can be accurately transmitted to all clients of the corresponding multicast group. Hence, it is reasonable for a user to join into several interesting multicast streams to receive video data.

Furthermore, in order to eliminate the additional network traffic arisen by the scheduling scheme, each stream is propagated at the video play rate. Clients use disks to store later played segments so that the received streams can be merged into an integrated stream.

3.2. Scheduling rules of the Medusa scheme

The objective of the Medusa scheme is to determine the frequency for scheduling the complete multicast streams so that the transmitting video data can be maximally shared among clients, and determine which segment will be transmitted on a patching multicast stream so that the amount of transmitted video data can be minimized. Notations used in this paper are showed in Table 1. Scheduling rules for the Medusa scheme are described as follows.

(1) The parallel video server dynamically schedules complete multicast streams. When the first request R_0 arrives, it schedules CS_0 at the end of time slot t_0 and notifies the corresponding clients of R_0 to receive and play back all segments transmitted on CS_0. Suppose the last complete multicast stream is CS_j $(0 \le j < +\infty)$. For an arbitrary client request R_i that arrives in the time

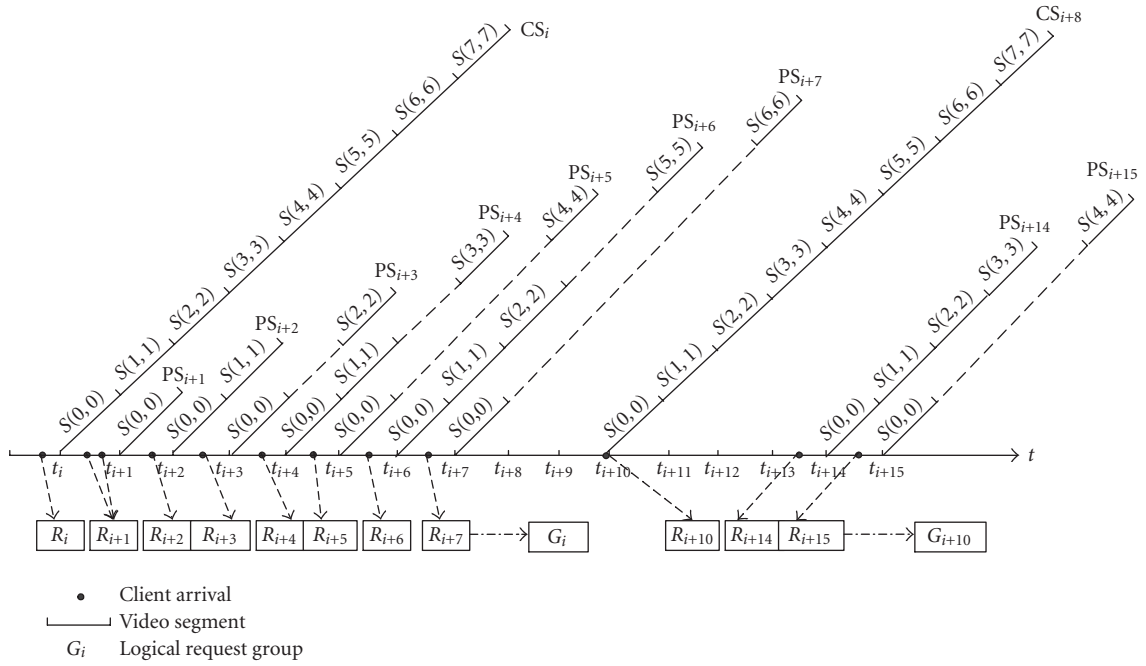


Figure 2: A scheduling example scene for the Medusa scheme.

t_i, if $t_j < t_i \leq t_j + \lceil L/T \rceil - 1$, no complete multicast stream need to be scheduled and just a patching multicast stream is scheduled according to rules (2) and (3). Otherwise, a new complete multicast stream CS$_i$ is initialized at the end of the time interval t_i.

(2) During the transmission of a complete multicast stream CS$_i$ ($0 \leq i < +\infty$), if a request R_j ($i < j \leq i + \lceil L/T \rceil - 1$) arrives, the server puts it into the logical requests group G_i. For each logical request group, a parallel video server maintains a stream information list. Each element of the stream information list records the necessary information of a patching multicast stream, described as a triple $E(t, I, A)$, where t is the scheduled time, I is the multicast group address of the corresponding patching multicast stream, and A is an array to record the serial number of video segments that will be transmitted on the patching multicast stream.

(3) For a client R_j whose request has been grouped into the logical group G_i ($0 \leq i < +\infty$, $i < j \leq i + \lceil L/T \rceil - 1$), the server notifies it to receive and buffer the later $\lceil L/T \rceil - (j - i)$ video segments from the complete multicast stream CS$_i$. Because the begining $j - i$ segments have been transmitted on the complete multicast stream CS$_i$, the client R_j loses them from CS$_i$. Thus, for each begining $j - i$ segments, the server searches the stream information list of G_i to find out which segment will be transmitted on an existing patching multicast stream and can be received by the client. If the lth segment ($0 \leq l < j - i$) will be transmitted on an existing patching multicast stream PS$_n$ ($i < n < j$) and the transmission start time is later than the service start time t_j, the server notifies the corresponding client R_j to join the multicast group of PS$_n$ to receive this segment. Otherwise, the server transmits this segment in a new initialized patching multicast stream PS$_j$ and notifies the client to join the multicast group of PS$_j$ to receive it. At last, the server creates the stream information element $E_j(t, I, A)$ of PS$_j$, and inserts it to the corresponding stream information list.

(4) Each multicast stream propagates the video data at the video playback rate. Thus, a video segment is completely transmitted during a time interval. For the mth segment that should be transmitted on the nth multicast stream, the start-transmission time is fixed and the value of this time can be calculated by the following equation:

$$\tau(m, n) = t_n + m^* T, \tag{1}$$

where t_n is the initial time of the nth multicast stream.

Figure 2 shows a scheduling example for the Medusa scheme. The requested video is divided into eight segments. Those segments are uniformly distributed on eight nodes in a round-robin fashion. The time unit on the t-axis corresponds to a time interval, as well as the total time it takes to deliver a video segment. The solid lines in the figure represent video segments transmitted on streams. The dotted lines show the amount of skipped video segments by the Medusa scheme.

In this figure, when the request R_i arrives at the time slot t_i, the server schedules a complete multicast stream CS$_i$.

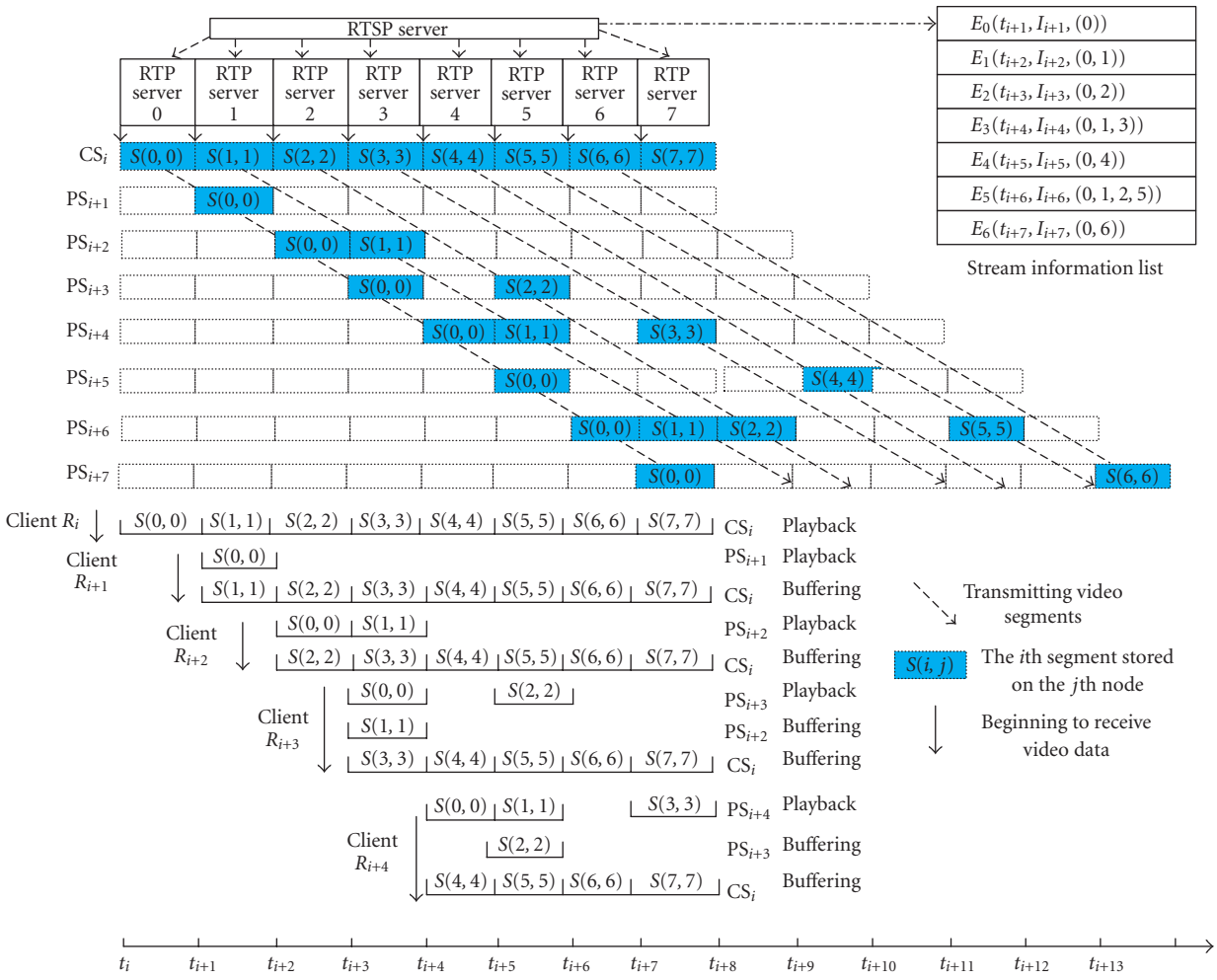


FIGURE 3: The scheduling course of parallel video server for requests of G_i and the corresponding receiving course for clients of R_i, R_{i+1}, R_{i+2}, R_{i+3}, and R_{i+4}.

Because the complete multicast stream is transmitted completely at time t_{i+10}, the video server schedules a new complete multicast stream CS_{i+10} to serve clients corresponding to the request R_{i+10}. According to rule (2), requests $R_{i+1} \cdots R_{i+7}$ must be grouped into the logical request group G_i, and requests R_{i+14}, R_{i+15} must be grouped into logical requests group G_{i+10}.

The top half portion of Figure 3 shows the scheduling of parallel video servers for those requests in the group G_i presented in Figure 2. The bottom half portion of Figure 3 shows the video data receiving and the stream-merging for clients R_i, R_{i+1}, R_{i+2}, R_{i+3}, and R_{i+4}. We just explain the scheduling for the request R_{i+4}, others can be deduced by rule (3). When request R_{i+4} arrives, the parallel video server firstly notifies the corresponding clients of R_{i+4} to receive the video segments $S(4,4)$, $S(5,5)$, $S(6,6)$, and $S(7,7)$ from the complete multicast stream CS_i. It searches the stream information list, and finds out that segment $S(2,2)$ will be transmitted on patching multicast stream PS_{i+3} and the transmission start time of $S(2,2)$ is later than t_{i+4}. Then, it notifies the client R_{i+4} to receive the segment $S(2,2)$ from patching multicast stream PS_{i+3}. At last, the parallel video server schedules a new patching multicast stream PS_{i+4} to transmit the missing segments $S(0,0)$, $S(1,1)$, and $S(3,3)$. The client R_{i+4} is notified to receive and play back those missing segments and the stream information element of PS_{i+4} is inserted into the stream information list.

4. DETERMINISTIC TIME INTERVAL

The value of time interval T is the key issue affecting the performance of the parallel video servers. In the Medusa scheme, a client may receive several multicast streams concurrently and the number of concurrently received multicast streams is related with the value of T. If T is too small, the number of concurrently received streams may be increased dramatically and exceed the client bandwidth capacity b_c. Some valid video data may be discarded at the client side. Furthermore, since a small T would increase the number of streams sent by the parallel video server, the server bandwidth efficiency may

be decreased. If T is too large, the startup latency may be too long to be endured and the client reneging probability may be increased.

In this section, we derive the deterministic time interval T which guarantee the startup latency minimized under the condition that the number of streams concurrently received by a client would not exceed the client bandwidth capacity b_c. The server bandwidth consumption affected by the time interval will be studied in Section 6.

We first derive the relationship between the value of $\mathrm{PB}_{\max}$ (defined in Table 1) and the value of T. For a request group G_i $(0 \le i < +\infty)$, CS_i is the complete multicast stream scheduled for serving the requests of G_i. For a request R_k $(i < k \le \lceil L/T \rceil - 1 + i)$ belonging to G_i, the clients corresponding to the R_k may concurrently receive several patching multicast streams. Assume that PS_j $(i < j < k)$ is the first patching stream from which clients of R_k can receive some video segments. According to the Medusa scheme, video segments from the $(j-i)$th segment to the $(\lceil L/T \rceil - 1)$th segment would not be transmitted on PS_j, and the $(j-i-1)$th segment would not be transmitted on the patching multicast streams initialized before the initial time of PS_j. Hence, the $(j-i-1)$th segment is the last transmitted segment for PS_j. According to (1), the start time for transporting the $(j-i-1)$th segment on PS_j can be expressed by

$$\tau(j-i-1, j) = t_j + (j-i-1)^* T. \tag{2}$$

Since the clients of R_k receive some video segments from PS_j, the start transporting time of the last segment transmitted on PS_j must be later than or equal to the request arrival time t_k. Therefore, we can obtain that

$$\tau(j-i-1, j) \ge t_k. \tag{3}$$

Consider that $t_k = t_j + (k-j) \times T$. Combining (2) and (3), we derive that

$$j \ge \frac{k+i+1}{2}. \tag{4}$$

If the clients of the request R_k receive some segments from the patching multicast streams $\mathrm{PS}_j, \mathrm{PS}_{j+1}, \ldots, \mathrm{PS}_{k-1}$, the number of concurrently received streams access to its maximum value. Thus, $\mathrm{PB}_{\max} = k - j$. Combing (4), we can obtain that $\mathrm{PB}_{\max} \le (k-i-1)/2$. In addition, because the request R_k belongs the request group G_i, the value of k must be less than or equal to $i + \lceil L/T \rceil - 1$, where L is the total playback time of the requested video object. Thus, $\mathrm{PB}_{\max}$ can be expressed by

$$\mathrm{PB}_{\max} = \left\lceil \frac{L}{2T} \right\rceil - 1. \tag{5}$$

For guaranteeing that the video data would not be discarded at the client end, the client bandwidth capacity must be larger than or equal to the maximum number of streams concurrently received by a client. It means that $b_c \ge \mathrm{PB}_{\max} + 1$, where 1 is the number of complete multicast streams received by a client. Combing (5), we obtain that

$$b_c \ge \left\lceil \frac{L}{2T} \right\rceil \Longrightarrow T \ge \left\lceil \frac{L}{2b_c} \right\rceil. \tag{6}$$

Obviously, the smaller the time interval T, the shorter the startup latency. Thus, the deterministic time interval will be the minimum value of T, that is,

$$T = \left\lceil \frac{L}{2b_c} \right\rceil. \tag{7}$$

5. PERFORMANCE EVALUATION

We evaluate the performance of the Medusa scheme via two methods: the mathematical analysis on the required server bandwidth, and the experiment. Firstly, we analyze the server bandwidth requirement for one video object in the Medusa scheme and compare it with the FCFS batching scheme and the stream-merging schemes. Then, the experiment for evaluating and comparing the performance of the Medusa scheme, the batching scheme, and the stream-merging schemes will be presented in detail.

5.1. Analysis for the required server bandwidth

Assume that requests for the ith video object are generated by a Poisson process with mean request rate λ_i. For serving requests that are grouped into the group G_j, the patching multicast streams $\mathrm{PS}_{j+1}, \mathrm{PS}_{j+2}, \ldots, \mathrm{PS}_{j+\lceil L/T \rceil - 1}$ may be scheduled from time t_{j+1} to time $t_{j+\lceil L/T \rceil - 1}$, where L is the length of the ith video object and T is the selected time interval. We use the matrix M_{pro} to describe the probabilities of different segments transmitted on different patching multicast streams. It can be expressed as

$$M_{\mathrm{pro}} = \begin{bmatrix} P_{11} & \cdots & P_{1(\lceil L/T \rceil - 1)} \\ P_{21} & \cdots & P_{2(\lceil L/T \rceil - 1)} \\ \vdots & \vdots & \vdots \\ P_{(\lceil L/T \rceil - 1)1} & \cdots & P_{(\lceil L/T \rceil - 1)(\lceil L/T \rceil - 1)} \end{bmatrix}, \tag{8}$$

where the nth column represents the nth video segment, the mth row expresses the patching multicast stream PS_{j+m}, and P_{mn} describes the probability for transmitting the nth segment on the patching multicast stream PS_{j+m} $(1 \le m \le \lceil L/T \rceil - 1,\ 1 \le n \le \lceil L/T \rceil - 1)$. Hence, the expected amount (in bits) of video data transmitted for serving requests grouped in G_j can be expressed as

$$\overline{\Omega} = b \times T \times \sum_{m=1}^{\lceil L/T \rceil - 1} \sum_{n=1}^{\lceil L/T \rceil - 1} P_{mn} + b \times L, \tag{9}$$

where b is the video transporting rate (i.e., the video playback rate) and $b \times L$ represents the number of video segments transmitted on the completely multicast stream CS_j.

According to the scheduling rules of the Medusa scheme, the nth $(1 < n \le \lceil L/T \rceil - 1)$ video segment should not be transmitted on patching multicast streams $\mathrm{PS}_{j+1}, \ldots, \mathrm{PS}_{j+n-1}$. Thus,

$$P_{mn} = 0 \quad \text{if } n > m. \tag{10}$$

On one hand, for the mth patching multicast stream, the first video segment and the mth video segment must be transmitted on it. This is because the first video segment has been transmitted completely on the patching multicast streams $\mathrm{PS}_{j+1}, \ldots, \mathrm{PS}_{j+m-1}$, and the mth video segment is not transmitted on such streams. We can obtain that P_{m1} and P_{mm} are equal to the probability for scheduling PS_{j+m} (i.e., the probability for some requests arriving in the time slot t_{j+m}). Since the requests for the ith video object are generated by Poisson process, the probability for some requests arriving in the time slot t_{j+m} can be calculated by $P[K \neq 0] = 1 - P[K = 0] = 1 - e^{-\lambda_i T}$. Considering that probabilities for request arriving in different time slots are independent from each other, we can derive that

$$\begin{aligned} P_{11} &= P_{21} = \cdots = P_{(\lceil L/T \rceil - 1)1} = P_{22} \\ &= P_{33} = \cdots = P_{(\lceil L/T \rceil - 1)(\lceil L/T \rceil - 1)} = 1 - e^{-\lambda_i T}. \end{aligned} \tag{11}$$

On the other hand, if the nth video segment is not transmitted on patching multicast streams from $\mathrm{PS}_{j+m-n+1}$ to PS_{j+m-1}, it should be transmitted on the patching multicast stream PS_{j+m}. Therefore, the probability for transmitting the nth segment on the mth patching multicast stream can be expressed as

$$\begin{aligned} P_{mn} = P_{m1} \times \prod_{k=m-n+1}^{m-1} (1 - P_{kn}) \\ (1 < n < m \leq \lceil L/T \rceil - 1), \end{aligned} \tag{12}$$

where P_{m1} represent the probability for scheduling the patching multicast stream PS_{j+m}, and $\prod_{k=m-n+1}^{m-1}(1 - P_{kn})$ indicates the probability for which the nth video segments would not be transmitted on patching multicast streams from $\mathrm{PS}_{j+m-n+1}$ to PS_{j+m-1}. Combining (9), (10), (11), and (12), we derive that

$$\begin{aligned} \overline{\Omega} = b \times T \times (1 - e^{-\lambda_i T}) \\ \times \sum_{m=1}^{\lceil L/T \rceil - 1} \sum_{n=1}^{m} \prod_{k=m-n+1}^{m-1} (1 - P_{kn}) + b \times L, \end{aligned} \tag{13}$$

where P_{kn} can be calculated by the following equations:

$$P_{kn} = \begin{cases} 0 & \text{if } k < n, \\ 1 - e^{-\lambda_i T} & \text{if } k = n, \\ \prod_{\ell=k-n+1}^{k-1} (1 - P_{\ell n}) & \text{if } k > n. \end{cases} \tag{14}$$

Because the mean number of arrived clients in the group G_j is $L \times \lambda_i$, we can derive that, in the time epoch $[t_j, t_{j+\lceil L/T \rceil - 1})$, the average amount of transmitted video data for a client, denoted by β_c, is

$$\begin{aligned} \beta_c &= \frac{\overline{\Omega}}{L \times \lambda_i} \\ &= \frac{b \times T \times (1 - e^{-\lambda_i T}) \sum_{m=1}^{\lceil L/T \rceil - 1} \sum_{n=1}^{m} \prod_{k=m-n+1}^{m-1} (1 - P_{kn})}{L \times \lambda_i} \\ &\quad + \frac{b}{\lambda_i}, \end{aligned} \tag{15}$$

where P_{kn} can be calculated by (14).

Consider the general case from time 0 to t. We derive the required average server bandwidth by modeling the system as a renewal process. We are interested in the process $\{B(t) : t > 0\}$, where $B(t)$ is the total server bandwidth used from time 0 to t. In particular, we are interested in the average server bandwidth $B_{\text{average}} = \lim_{t \to \infty} S(t)/t$. Let $\{t_j \mid (0 \leq j < \infty), (t_0 = 0)\}$ denote the time set for a parallel video server to schedule a complete multicast stream for video i. These are renewal points, and the behavior of the server for $t \geq t_j$ does not depend on past behavior. We consider the process $\{B_j, N_j\}$, where B_j denotes the total server bandwidth consumed and N_j denotes the total number of clients served during the jth renewal epoch $[t_{j-1}, t_j)$. Because this is a renewal process, we drop the subscript j and have the following result:

$$B_{\text{average}} = \lambda_i \frac{E[B]}{E[N]}. \tag{16}$$

Obviously, $E[N] = \lambda_i \times L$. For $E[B]$, let K denote the number of arrivals in an interval of renewal epoch length L. It has the distribution $P[K = \kappa] = (\lambda_i \times L)^{\kappa} e^{-\lambda_i L}/\kappa!$. For $E[B \mid K = \kappa]$, we have

$$\begin{aligned} E[B \mid K = \kappa] &= \kappa \beta_c \\ &= \left(\frac{b \times T \times (1 - e^{-\lambda_i T}) \sum_{m=1}^{\lceil L/T \rceil - 1} \sum_{n=1}^{m} \prod_{k=m-n+1}^{m-1} (1 - P_{kn})}{L \times \lambda_i} \right. \\ &\quad \left. + \frac{b}{\lambda_i} \right) \kappa. \end{aligned} \tag{17}$$

This indicates that κ Poisson arrivals in an interval of length L are equally likely to occur anywhere within the interval. Removal of the condition yields

$$E[B] = \sum_{\kappa=1}^{\infty} \frac{(\lambda_i \times L)^{\kappa} e^{-\lambda_i L}}{\kappa!} E[B \mid K = \kappa]. \tag{18}$$

Combining (17) and (18), we derive that

$$\begin{aligned} E[B] = b \times T \times (1 - e^{-\lambda_i T}) \\ \times \sum_{m=1}^{\lceil L/T \rceil - 1} \sum_{n=1}^{m} \prod_{k=m-n+1}^{m-1} (1 - P_{kn}) + b \times L. \end{aligned} \tag{19}$$

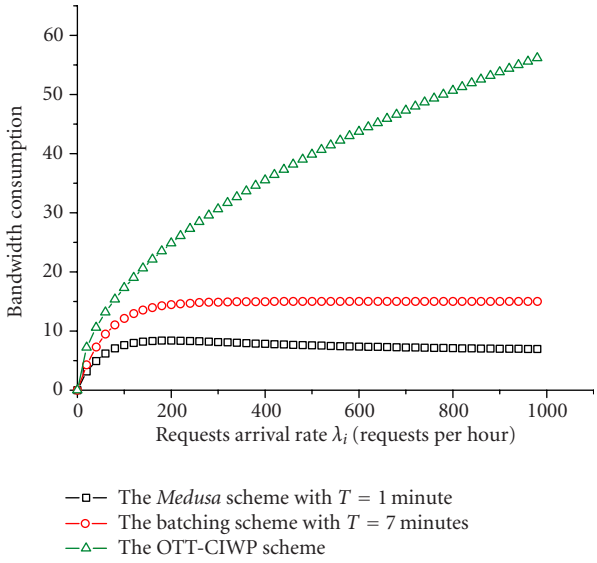


Figure 4: Comparison of the expected server bandwidth consumption for one video object among the Medusa scheme, the batching scheme, and the OTT-CIWP scheme.

According to (16) and (19), we derive that

$$B_{\text{average}} = b \times T \times \left(1 - e^{-\lambda_i T}\right) \times \sum_{m=1}^{\lceil L/T \rceil - 1} \sum_{n=1}^{m} \prod_{k=m-n+1}^{m-1} \frac{1 - P_{kn}}{L} + b. \tag{20}$$

For the batching schemes, since all scheduled streams are completely multicast streams, the required server bandwidth for the ith video object can be expressed as

$$B_{\text{average}}(\text{batching}) = b \times \left(1 - e^{-\lambda_i T}\right) \times \left\lceil \frac{L}{T} \right\rceil. \tag{21}$$

For the stream merging schemes, we choose the optimal time-threshold CIWP (OTT-CIWP) scheme for comparison. Gao and Towsley [20] have showed that the OTT-CIWP scheme outperforms most other stream-merging schemes and the required server bandwidth for the ith video object has been derived as

$$B_{\text{average}}(\text{OTT - CIWP}) = b \times \left(\sqrt{2L\lambda_i + 1} - 1\right). \tag{22}$$

Figure 4 shows the numerical results for comparing the required server bandwidth of one video object among the Medusa scheme, the batching scheme, and the OTT-CIWP scheme. In Figure 4, the chosen time interval T for the Medusa scheme is 1 minute while the batching time threshold for the batching scheme is 7 minutes. In addition, the length of the ith video object is 100 minutes. As one can see, the Medusa scheme significantly outperforms the batching scheme and the OTT-CIWP scheme over a wider range of request arrival rate.

5.2. Experiment

In order to evaluate the performance of the Medusa scheme in the general case that multiple video objects of varying popularity are stored on the parallel video servers, we use the *Turbogrid* streaming server[1] with 8 RTP server nodes as the experimental platform.

5.2.1. Experiment environment

We need two factors for each video, its length and its popularity. For its length, the data from the Internet Movie Database (http://www.imdb.com) has shown a normal distribution with a mean of 102 minutes and a standard deviation of 16 minutes. For its popularity, Zipf-like distribution [21] is widely used to describe the popularity of different video objects. Empirical evidence suggests that the parameter θ of the Zipf-like distribution is 0.271 to give a good fit to real video rental [5, 6]. It means that

$$\pi_i = \frac{1}{i^{0.729}} \left[\sum_{k=1}^{N} \frac{1}{k^{0.729}} \right], \tag{23}$$

where π_i represents the popularity of the ith video object and N is the number of video objects stored on the parallel video servers.

Client requests are generated using a Poisson arrival process with an interval time of $1/\lambda$ for varying λ values between 200 to 1600 arrivals per hour. Once generated, clients simply select a video and wait for their request to be served. The waiting tolerance of each client is independent of the other, and each is willing to wait for a period time $U \geq 0$ minutes. If its requested movie is not displayed by then, it reneges. (Note that even if the start time of a video is known, a client may lose its interest in the video and cancel its request. If it is delayed too long, in this case, the client is defined "reneged.") We consider the exponential reneging function $R(u)$, which is used by most VOD studies [6, 7, 15]. Clients are always willing to wait for a minimum time $U_{\min} \geq 0$. The additional waiting time beyond $U_{\min}$ is exponentially distributed with mean τ minutes, that is,

$$R(u) = \begin{cases} 0 & \text{if } 0 \leq u \leq U_{\min} \\ 1 - e^{-(u - U_{\min})/\tau}, & \text{otherwise.} \end{cases} \tag{24}$$

Obviously, the larger τ is, the more delay clients can tolerate. We choose $U_{\min} = 0$ and $\tau = 15$ minutes in our experiment. If the client is not reneging, it simply plays back the received streams until those streams are transmitted completely.

Considering that the popular set-top boxes have similar components (CPU, disk, memory, NIC, and the dedicated client software for VOD services) to those of PCs, we use PCs to simulate the set-top boxes in our experiment. In addition, because the disk is cheaper, faster, and bigger than ever, we

[1] Turbogrid streaming server is developed by the Internet and Cluster Computing Center of Huazhong University of Science and Technology.

Table 2: Experiment parameters.

Video length (min) L	90 ~ 120
Number of videos Nv	200
Video format	MPEG-1
Clients' bandwidth capacity (Mbits/s)	100
Maximum total bandwidth of parallel video server (Mbits/s)	1000
Clients arrival rate λ (hour)	200 ~ 1600

do not consider the speed limitation and the space limitation of disk. Table 2 shows the main experimental environment parameters.

5.2.2. Results

For parallel video servers, there are two most important performance factors. One is startup latency, which is the amount of time clients must wait to watch the demanded video, the other is the average bandwidth consumption, which indicates the bandwidth efficiency of the parallel video servers. We will discuss our results in these two factors.

(A) *Startup latency and reneging probability*

As discussed in Section 4, in order to guarantee that clients can receive all segments of their requested video objects, the minimum value of time interval (i.e., optimal time interval) T will be $\lceil L/(2b_c) \rceil \cong 120/2^*60 = 1$ minute. We choose time interval T to be 1, 5, 10, and 15 minutes for studying the effect on the average startup latency and the reneging probability, respectively. Figures 5 and 6 display the experimental results at these two factors. By the increase of time interval T, the average startup latency and the reneging probability are also increased. When T is equal to the deterministic time interval $T = 1$ minute, the average startup latency is less than 45 seconds and the average reneging probability is less than 5%. But when T is equal to 15 minutes, the average startup latency is increased to near 750 seconds and almost 45% of clients renege. Figures 7 and 8 display a startup latency comparison and a reneging probability comparison among the FCFS batching scheme with time interval $T = 7$ minutes, and the OTT-CIWP scheme [20] and the Medusa scheme with deterministic time interval $T = 1$ minute. We choose 7 minutes because [7] has presented that FCFS batching could obtain a good trade-off between startup latency and bandwidth efficiency at this batching time threshold. As one can see, the Medusa scheme outperforms the FCFS scheme and is just little poorer than the OTT-CIWP scheme at the aspect of the system average startup latency and reneging probability. The reason for this little poor performance compared with OTT-CIWP is that the Medusa scheme batches client requests arriving in the same time slot. This will effectively increase the bandwidth efficiency at high client-request rates.

(B) *Bandwidth consumption*

Figure 9 shows how the time interval T affects the server's average bandwidth consumption. We find out that the server's

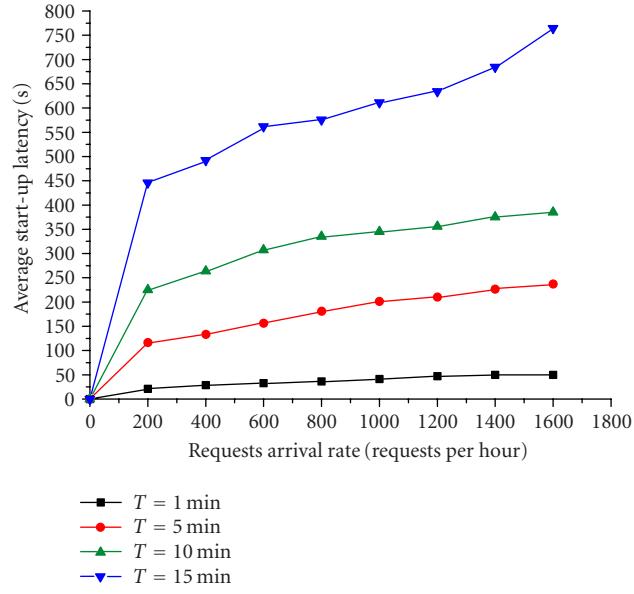


Figure 5: The effect of time interval T on the average startup latency.

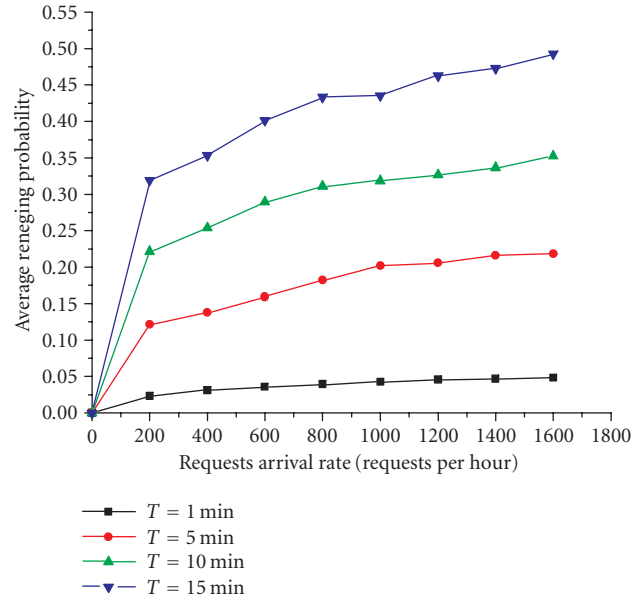


Figure 6: The effect of time interval T on the average reneging probabity.

average bandwidth consumption is decreased by some degree by increasing the time interval. The reason is that more clients are batched together and served as one client. Also, we can find out that the decreasing degree of bandwidth consumption is very small when client-request arrival rate is less than 600 requests per hour. When the arrival rate is more than 600, the decreasing degree tends to be distinct. However, when the request arrival rate is less than 1600 requests

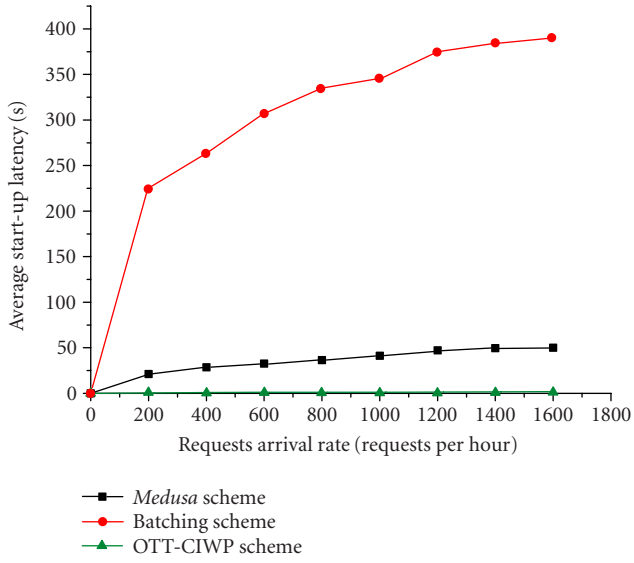


Figure 7: A startup latency comparison among the batching scheme with time interval $T = 7$ minutes, the OTT-CIWP scheme, and Medusa scheme with time interval $T = 1$ minute.

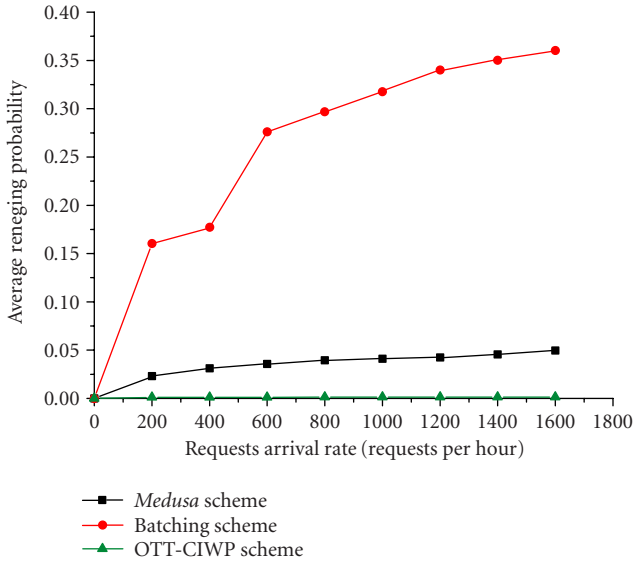


Figure 8: A reneging probability comparison among the batching scheme with time interval $T = 7$ minutes, the OTT-CIWP scheme, and the Medusa scheme with time interval $T = 1$ minute.

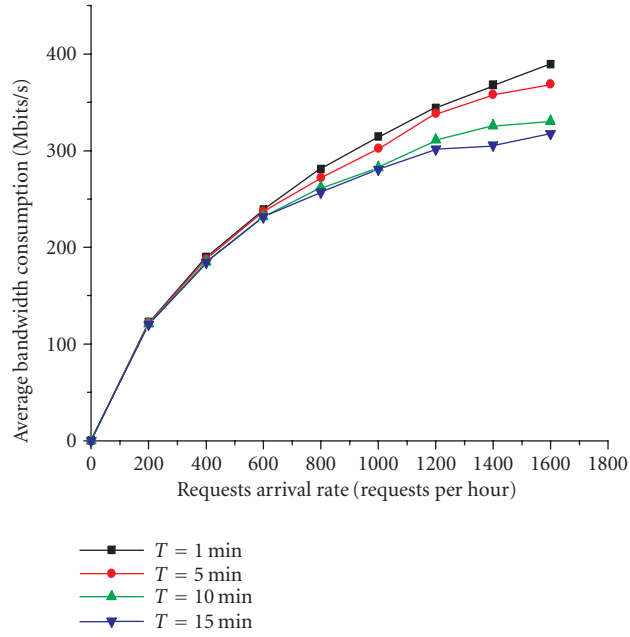


Figure 9: How time interval T affects the average bandwidth consumption.

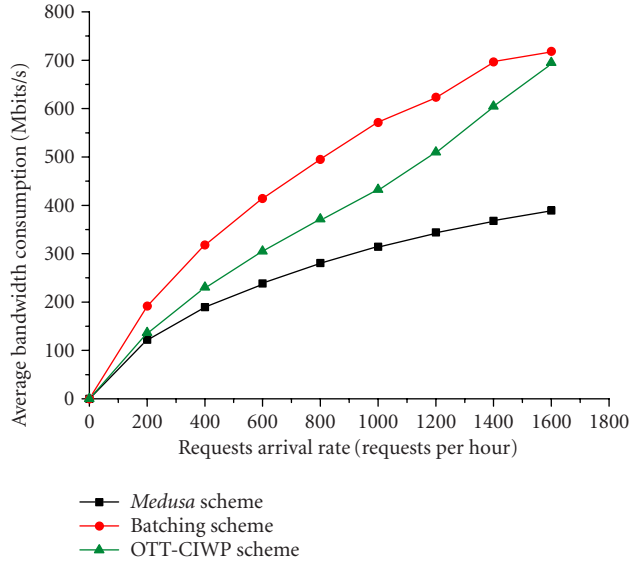


Figure 10: Average bandwidth consumption comparison among the batching scheme with time interval $T = 7$ minutes, the OTT-CIWP scheme, and the Medusa scheme with time interval $T = 1$ minute.

per hour, the total saved bandwidth is less than 75 Mbits/s by comparing deterministic time intervals $T = 1$ minute and $T = 15$ minutes. On the other hand, the clients reneging probability is dramatically increased form 4.5% to 45%. Therefore, a big time interval T is not a good choice and we suggest using $\lceil L/(2b_c) \rceil$ to be the chosen time interval.

As showed on Figure 10, when the request arrival rate is less than 200 requests per hour, the bandwidth consumption of three kinds of scheduling strategies are held in the same level. But by increasing the request-arrival rate, the bandwidth consumption increasing degree of the Medusa scheme is distinctly less than that of the FCFS batching and the OTT-CIWP. When the request-arrival rate is 800, the average bandwidth consumption of the Medusa scheme is approximately 280 Mbits/s. At the same request-arrival rate, the average bandwidth consumption of the FCFS batching is

approximately 495 Mbits per second and that of the OTT-CIWP is approximately 371 Mbits per second. It indicates that, at middle request-arrival rate, the Medusa scheme can save approximately 45% bandwidth consumption compared with FCFS batching, and can save approximately 25% bandwidth consumption compared with OTT-CIWP.

When the request arrival rate is higher than 1500 requests per hour, the bandwidth performance of OTT-CIWP is going to be worse and worse. It is near to the FCFS batching scheme. In any case, the Medusa scheme significantly outperforms the FCFS scheme and the OTT-CIWP scheme. For example, as shown in Figure 10, the Medusa scheme just consumes 389 Mbits/s server bandwidth at the request arrival rate 1600 requests per hour, while the FCFS batching scheme consumes 718 Mbits/s server bandwidth and the OTT-CIWP scheme needs 694 Mbits/s. Therefore, we can conclude that the Medusa scheme is distinctly outperforming the batching scheme and the OTT-CIWP scheme at the aspect of bandwidth performance.

6. DISCUSSIONS

For the Medusa scheme, two issues must be considered carefully, the client network architecture and the segments placement policy. In this section, we give out some discussions on the effect of these two issues.

6.1. Homogenous client network versus heterogeneous client network

In the above discussions, we discuss the homogenous client network based on the FTTB network architecture. If the parallel video servers are used for serving the VOD systems with heterogeneous client network architecture such as the cable modem access and 10 M LAN access, the basic Medusa scheme is not recommended. This is because the small client bandwidth capacity would result in a large deterministic time interval T, as well as the long startup latency and the high reneging probability. However, we can extend the Medusa scheme as following for the heterogeneous client network.

For cable modem users, because the client bandwidth capacity is lower than 2 Mb per second, it just has the capacity to receive one MPEG-I stream (approximately $1.2 \sim 1.5$ Mb per second per stream). In this case, the stream merging schemes and the Medusa scheme are not suitable. We use the batching scheme to schedule streams. Note that the client bandwidth capacity is sent to the parallel video servers during the session being in setup. Thus, the parallel video server can distinguish the category of clients before determining how to schedule streams for serving them.

For 10 M LAN users, the client bandwidth capacity is enough to concurrently receive near 6 MPEG-I streams. In this case, if we use the basic Medusa scheme, the deterministic time interval for a video object with 120 minutes length is 10 minutes and the expected startup latency is near 5 minutes. It is too long for most clients. However, we can extend the basic Medusa scheme to use a time window W to control the scheduling frequency of the complete multicast streams. If requests arrive in the same time window, the parallel video server schedules patching multicast streams according to the basic Medusa scheduling rule (3). Otherwise, a new complete multicast stream will be scheduled. According to the deriving course discussed in Section 4, we can easily obtain that the deterministic time interval T should be $\lceil W/(2b_c) \rceil$. Obviously, if the value of time window W is smaller than the length of the requested video object, the deterministic time interval T and the expected startup latency can be decreased. However, a small time window W would increase the required server bandwidth. The detailed relationship between the time window W, the expected startup latency, and the required server bandwidth will be studied in our further works.

6.2. Effect of the segment placement policy

For the scheduling of the Medusa scheme, the begining segments of a requested video are transmitted more frequently than the later segments of that video. It is called *intra-movie skewness* [22]. If segments of all stored videos are distributed from the first RTP server node to the last RTP server node in a round-robin fashion, the intra-movie skewness would result in that the load for the first RTP server node is far heavier than the load of other RTP server nodes so that the load balance of parallel video servers is destroyed.

Two kinds of segments placement policies were proposed to solve the intra-movie skewness problem: the symmetric pair policy [22, 23] and the random policy.

In the symmetric pair policy, based on the serial number of video objects, all stored video objects are divided into two video sequences, the odd sequence and the even sequence. For the odd video sequence, the jth segment of the ith video object ($i = 1, 3, 5, \ldots, 2k + 1$) is located on the $((2^*N - 1 - (j + (i/2)) \bmod N) \bmod N)$th RTP server node, where N is the total number of RTP server nodes. For the even video sequence, the jth segment of the ith video object ($i = 0, 2, 4, \ldots, 2k$) is located on the $((j + (i/2)) \bmod N)$th RTP server node. As discussed in [22, 23], these placement rules can uniformly distribute segments with high transmission frequency to different RTP server nodes so that the load balance of the parallel video server can be guaranteed.

The random placement policy randomly distributes video segments on different RTP server nodes so that the probabilistic guarantee of load balancing can be provided. Santos et al. [24] have shown that the random placement policy has better adaptability to different user access patterns and can support more generic workloads than the symmetric pair policy. For load balancing performance, these two schemes have very similar balancing results [24]. However, the random placement scheme only provides probabilistic guarantee of load balancing and it has the drawback of maintaining a huge video index of the striping data blocks. Hence, we use the symmetric pair policy to solve the load balancing problem in the Medusa scheme.

7. CONCLUSIONS AND FUTURE WORKS

In this paper, we focus on the homogenous FTTB client network architecture and propose a novel stream-scheduling

scheme that significantly reduces the demand on the server network I/O bandwidth of parallel video servers. Unlike existing batching scheme and stream-merging scheme, the Medusa scheme dynamically groups the clients' requests according to their request arrival time and schedules two kinds of multicast streams, the completely multicast stream and the patching multicast stream.

For the clients served by patching multicast streams, the Medusa scheme notifies them to receive the segments that will be transmitted by other existing patching multicast streams and only transmit the missed segments on the new scheduled stream. This guarantees that no redundant video data are transmitted at the same time period and that the transmitting video data are shared among grouped clients. The mathematical analysis and the experiment results show that the performance of the Medusa scheme significantly outperforms the batching schemes and the stream-merging schemes.

Our ongoing research includes

(1) designing and analyzing the *extended-Medusa* scheme for clients with heterogeneous receive bandwidths and storage capacities,
(2) evaluating the impact of VCR operations on the required server bandwidth for the Medusa scheme,
(3) developing optimized caching models and strategies for the Medusa scheme,
(4) designing optimal real-time delivery techniques that support recovery from packet loss.

ACKNOWLEDGMENT

This paper was supported by the National Hi-Tech Project under Grant 2002AA1Z2102.

REFERENCES

[1] C. Shahabi, R. Zimmermann, K. Fu, and S.-Y. D. Yao, "Yima: a second generation continuous media server," *IEEE Computer magazine*, vol. 35, no. 6, pp. 56–64, 2002.
[2] G. Tan, H. Jin, and L. Pang, "A scalable video server using intelligent network attached storage," in *Management of Multimedia on the Internet: 5th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, vol. 2496 of *Lecture Notes in Computer Sciences*, pp. 114–126, Santa Barbara, Calif, USA, October 2002.
[3] G. Tan, H. Jin, and S. Wu, "Clustered multimedia servers: architectures and storage systems," in *Annual Review of Scalable Computing*, vol. 5, pp. 92–132, World Scientific, Singapore, 2003.
[4] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On optional batching policies for video-on-demand storage servers," in *Proc. 3rd IEEE International Conference on Multimedia Computing and Systems*, pp. 312–316, Hiroshima, Japan, June 1996.
[5] S. W. Carter and D. D. E. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. 6th International Conference on Computer Communication and Networks*, pp. 200–207, Las Vegas, Nev, USA, September 1997.
[6] S.-H. G. Chan and F. Tobagi, "Tradeoff between system profit and user delay/loss in providing near video-on-demand service," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 8, pp. 916–927, 2001.
[7] J.-K. Chen and J.-L. C. Wu, "Heuristic batching policies for video-on-demand services," *Computer Communications*, vol. 22, no. 13, pp. 1198–1205, 1999.
[8] D. L. Eager and M. K. Vernon, "Dynamic skyscraper broadcasts for video-on-demand," Tech. Rep. 1375, Department of Computer Science, University of Wisconsin, Madison, Wis, USA, 1998.
[9] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "The maximum factor queue length batching scheme for video-on-demand systems," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 97–110, 2001.
[10] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proc. 2nd ACM International Conference on Multimedia*, pp. 15–23, San Francisco, Calif, USA, October 1994.
[11] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, no. 3, pp. 112–121, 1996.
[12] H. J. Kim and Y. Zhu, "Channel allocation problem in VOD system using both batching and adaptive piggybacking," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 969–976, 1998.
[13] S. W. Carter and D. D. E. Long, "Improving bandwidth efficiency on video-on-demand servers," *Computer Networks*, vol. 30, no. 1-2, pp. 99–111, 1999.
[14] S.-H. G. Chan and E. Chang, "Providing scalable on-demand interactive video services by means of client buffering," in *Proc. IEEE International Conference on Communications*, pp. 1607–1611, Helsinki, Finland, June 2001.
[15] D. Eager, M. Vernon, and J. Zahorjan, "Bandwidth skimming: A technique for cost-effective video-on-demand," in *Proc. Multimedia Computing and Networking 2000*, San Jose, Calif, USA, January 2000.
[16] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. 6th ACM International Multimedia Conference*, pp. 191–200, Bristol, UK, September 1998.
[17] W. Liao and V. O. K. Li, "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia*, vol. 4, no. 4, pp. 51–62, 1997.
[18] J.-F. Paris, S. W. Carter, and D. D. E. Long, "A hybrid broadcasting protocol for video on demand," in *Proc. 1999 Multimedia Computing and Networking Conference*, pp. 317–326, San Jose, Calif, USA, January 1999.
[19] H. Shachnai and P. Yu, "Exploring wait tolerance in effective batching for video-on-demand scheduling," *Multimedia Systems*, vol. 6, no. 6, pp. 382–394, 1998.
[20] L. Gao and D. Towsley, "Threshold-based multicast for continuous media delivery," *IEEE Trans. Multimedia*, vol. 3, no. 4, pp. 405–414, 2001.
[21] G. Zipf, *Human Behavior and the Principle of Least Effort*, Addison Wesley, Boston, Mass, USA, 1949.
[22] S. Wu and H. Jin, "Symmetrical pair scheme: a load balancing strategy to solve intra-movie skewness for parallel video servers," in *International Parallel and Distributed Processing Symposium*, pp. 15–19, Fort Lauderdale, Fla, USA, April 2002.
[23] S. Wu, H. Jin, and G. Tan, "Analysis of load balancing issues caused by intra-movie skewness for parallel video servers," *Parallel and Distributed Computing Practices*, vol. 4, no. 4, pp. 451–465, 2003.
[24] J. Santos, R. Muntz, and B. Ribeiro-Neto, "Comparing random data allocation and data striping in multimedia servers," in *Proc. International Conference on Measurement and Modeling of Computer Systems*, pp. 44–55, Santa Clara, Calif, USA, June 2000.

Hai Jin is a Professor at the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China. He received M.S. and Ph.D. degrees at HUST in 1991 and 1994, respectively. He was a Postdoctoral Fellow at the Department of Electrical and Electronics Engineering, University of Hong Kong, and a visiting scholar at Department of Electrical Engineering-System, University of South California, Los Angeles, USA, from 1999 to 2000. His research interests include cluster computing, grid computing, multimedia systems, network storage, and network security. He is the Editor of several journals, such as *International Journal of Computers and Applications, International Journal of Grid and Utility Computing*, and *Journal of Computer Science and Technology*. He is now leading the largest grid project in China, called ChinaGrid, funded by the Ministry of Education, China.

Dafu Deng received his Bachelor degree in engineering from the Tongji University, Shanghai, China, in 1997, and is a Ph.D. candidate at the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China. His research interests include cluster computing, grid computing, multimedia systems, communication technologies, and P2P systems.

Liping Pang is a Professor at the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China. In 1995, she was awarded the "Golden medal in education of China." In the recent three years, she has over 40 publications and 3 books in computing science and education. Her research interests include parallel and distribution computing, grid computing, cluster computing, and multimedia technology.