

Abstract

Digital Video Transmission over Wireless Channels

Wireless technology has become the most exciting area in telecommunication and networking. The rapid growth of mobile telephone use, various satellite services, and now the wireless Internet are generating tremendous changes in telecommunications and networking. Wireless is convenient and often less expensive to deploy than fixed service, but wireless is not perfect. There are limitations, political and technical difficulties that may ultimately prevent wireless technologies from reaching their full potential. Regardless of the design of the transmission system, there will be errors, resulting in the change of one or more bits in a transmitted frame.

In this thesis, we focus on different characteristics of wireless channels and their applications to video transmission framework. Channel coding and interleaving techniques have long been recognized as an effective technique for combating the deleterious effects of noise, interference, jamming, fading, and other channel impairments. The basic idea of channel coding is to introduce controlled redundancy into the transmitted signals that is exploited at the receiver to correct channel induced errors by means of forward error correction. Channel coding can also be used for error detection in schemes that employ automatic repeat request (ARQ) strategies. ARQ strategies must have a feedback channel to relay the retransmission requests from the receiver back to the

transmitter when errors are detected. ARQ schemes require buffering at the transmitter and/or receiver and, therefore, are suitable for data applications but are not suitable for delay sensitive voice applications.

Problem of video transmission over wireless channels has been addressed in literature [5], [8], [10], [11], [12], [22], [23]. In this thesis, we will present a review of these techniques. However, we will mainly focus on different characteristics of wireless channels and their implications on video transmission framework. Therefore, before discussing specific tools in chapter 7, we first present the characteristics of wireless channels in chapter 5 and chapter 6. In chapter 8, we present the performance of the protocol, based on a network simulation. In chapter 9, we present conclusions and future work.

Contents

| | |
|--|-----------|
| List of Figures | v |
| List of Tables | vi |
| 1. Introduction..... | 1 |
| 1.1 Wireless LANs..... | |
| 2. Wireless Data Vision..... | 4 |
| 3. Current State..... | 6 |
| 4. Video Compression..... | 6 |
| 5. Wireless Channels..... | 13 |
| 6. Solutions to Wireless Specific Problems..... | 13 |
| 6.1 Forward Error Correction..... | 14 |
| 6.1.1 Media Independent FEC..... | 15 |
| 6.1.2 Congestion Control..... | 16 |
| 6.2 Interleaving..... | 18 |
| 6.3 Automatic Repeat ReQuest..... | 18 |
| 7. Tools for Digital Video Transmission on Wireless Channels..... | 21 |
| 7.1 Tools on Encoder Side..... | 23 |
| 7.1.1 Resynchronization Markers..... | 23 |
| 7.1.2 Concatenated FEC and Interleaving..... | 24 |

| | | |
|------------|---|-----------|
| 7.1.3 | Data Partitioning and Unequal Error Protection..... | 25 |
| 7.1.4 | Reversible Variable Length Codes..... | 27 |
| 7.1.5 | Joint Source-Channel Coding..... | 28 |
| 7.2 | Tools on Decoder Side..... | 31 |
| 7.2.1 | Error Detection..... | 31 |
| 7.2.2 | Soft In/ Soft Out Decoders..... | 33 |
| 7.2.3 | Temporal Extrapolation..... | 35 |
| 7.2.4 | Spatial Interpolation..... | 36 |
| 7.2.5 | Motion Vector Interpolation..... | 37 |
| 7.3 | Tools in the Existence of Reverse Channel..... | 40 |
| 7.3.1 | Automatic Repeat ReQuest..... | 41 |
| 7.3.2 | Error Tracking..... | 41 |
| 7.3.3 | Joint Source-Channel Coding..... | 49 |
| 8. | Visualization toolkit for simulating network in ns2..... | 52 |
| 8.1 | Network Simulator ns2..... | 52 |
| 8.2 | Emulator..... | 52 |
| 8.3 | Support software..... | 53 |
| 8.3.1 | Nam..... | 53 |
| 8.3.2 | XGraph..... | 54 |
| 8.4 | Performance..... | 55 |
| 9. | Program to demonstrate ARQ protocols | 59 |
| 9.1 | link.tcl | 59 |
| 9.2 | link_agent.tcl | 60 |
| 9.3 | timer.tcl | 62 |
| 10. | Conclusions..... | 64 |
| A | Reed-Solomon codes..... | 65 |
| A.1 | Reed-Solomon Encoder..... | 66 |
| A.2 | RS Codes for Binary Data..... | 68 |
| A.3 | Decoding of RS Codes..... | 68 |
| A.4 | Errors-only Decoding of RS Codes..... | 69 |
| | Reference..... | 70 |

List of Figures

| | | |
|----|---|----|
| 1 | An Architecture for Video Streaming..... | 3 |
| 2 | Example of MPEG GOP..... | 10 |
| 3 | Block Diagram for Motion-Compensated Hybrid Encoder..... | 12 |
| 4 | Interleaving..... | 18 |
| 5 | Block Diagram of Video over Wireless Channel..... | 22 |
| 6 | Encoding Procedures..... | 22 |
| 7 | Resynchronization Markers..... | 23 |
| 8 | Resynchronization Markers Present in Bit Stream..... | 24 |
| 9 | Concatenated FEC..... | 25 |
| 10 | Reversible Variable Length Codes..... | 28 |
| 11 | Decoding Procedures..... | 30 |
| 12 | Spatial Interpolation of Lost Pixels..... | 37 |
| 13 | A Simple Motion Vector Interpolation Method..... | 40 |
| 14 | Motion-Compensated Hybrid..... | 42 |
| 15 | Illustration of Spatiotemporal Error Propagation..... | 42 |
| 16 | Loss in SNR of the Decoded Video Signal..... | 44 |
| 17 | Illustration of Error Propagation when Error Tracking is Used..... | 45 |
| 18 | Error Recovery with Feedback Channel..... | 46 |
| 19 | Reconstructed Frames of Test Sequence..... | 48 |
| 20 | FGS Enhancement Layer Coding..... | 51 |
| 21 | System for H.263 and RCPC Encoder..... | 51 |
| 22 | Influence of the number of packet retransmissions on TCP throughput.... | 55 |
| 23 | TCP throughput using ARQ, FEC..... | 56 |

List of Tables

| | | |
|---|--|----|
| 1 | Typical MPEG-1 and MPEG-2 Coding Parameters..... | 8 |
| 2 | Motion Vector Entropy versus different Predictors..... | 39 |

1 Introduction

Guglielmo Marconi invented the wireless telegraph in 1896. In 1901, he sent telegraphic signals across the Atlantic Ocean from Cornwall to St. John's Newfoundland; a distance of 1800 miles. His invention allowed two parties to communicate by sending each other alphanumeric characters encoded in an analog signal. Over the last century, advances in wireless technologies have led to the radio, the television, the mobile telephone, and communication satellites. All types of information can now be sent to almost every corner of the world. Recently, a great deal of attention has been focused on satellite communication, wireless networking, and cellular technology.

Historically, wireless data communications was principally the domain of large companies with specialized needs. For example, large organizations need to stay in touch with their mobile sales force, or delivery services need to keep track of their vehicles and packages. However, this situation is steadily changing and wireless data communications is becoming as commonplace as its wired counterpart.

In recent years we have witnessed a persistent growth in wireless communication systems. Wireless access to the Internet may outstrip all other forms of access in the near future. It is likely that mobile users will expect similar levels of service quality as wireline users. One of the factors behind this growth was the dominance of digital wireless communication systems, which have superior bandwidths and can integrate voice and data communications. As a result, wireless communication systems now offer higher quality and quantity of services.

The need for wireless data communications arises partially because of the need for mobile computing and partially because of the need for specialized applications, such as computerized dispatch services and mobile fleet management.

Mobile computing, which aims to migrate the computing world onto a mobile environment, is affected primarily by two components: portability and connectivity. Portability, i.e., the ability to untether computers from the conventional desktop environment, is getting increasingly feasible because with the continuous improvement in integration, miniaturization, and battery technology, the differences in performance and cost between desktop and portable computers is shrinking. Therefore, the processing power of desktop computing is becoming available to portable environments and this is highly desirable as far as productivity is concerned.

In the last decade, there has also been a rapid development in the digital video compression field. Using new compression algorithms with high compression rates, digital video can provide superior quality with low bandwidths over its analog counterpart. With the addition of low and very-low bit rate coding algorithms, video transmission over wire-line channels is now a mature area.

Combination of advances in digital video compression and digital wireless communications resulted in a new service area called video over wireless. Potential applications such as HDTV and picture-phones attracted attentions, and now are under rapid development. Services related to video transmission over wireless channels are expected to grow exponentially in the near future.

Due to the proliferation of multimedia on WWW and the mergence of broadband wireless network, wireless video communication has received great interests from both

industry and academy. However, the characteristics of wireless channel impose new problems to video transmission.

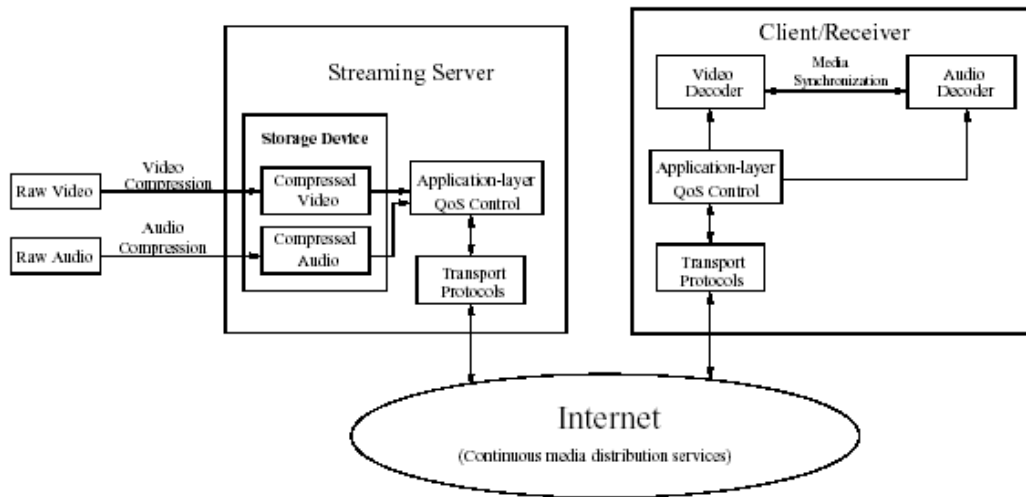


Figure 1: An architecture for video streaming

1.1 Wireless LANs

Wireless LANs provide high-speed data within a small region, e.g. a campus or small building, as users move from place to place. Wireless devices that access these LANs are typically stationary or moving at pedestrian speeds. Nearly all wireless LANs in the United States use one of the ISM frequency bands. The appeal of these frequency bands, located at 900 MHz, 2.4 GHz, and 5.8 GHz, is that an FCC license is not required to operate in these bands. However, this advantage is a double-edged sword, since many other systems operate in these bands for the same reason, causing a great deal of interference between systems. The FCC mitigates this interference problem by setting a stringent limit on the power per unit bandwidth for ISM-band systems. To satisfy this requirement wireless LANs use either direct sequence or frequency hopping spread spectrum so that their total power is spread over a wide bandwidth. Wireless LANs can

have either a star architecture, with wireless access points or hubs placed throughout the coverage region, or a peer-to-peer architecture, where the wireless terminals self-configure into a network.

2 Wireless Data Vision

The vision of wireless communications providing high-speed high-quality information exchange between portable devices located anywhere in the world is the communications frontier of the next century. This vision will allow people to operate a virtual office anywhere in the world using a small handheld device with seamless telephone, modem, fax, and computer communications. Wireless LANs will be used to connect together palmtop, laptop, and desktop computers anywhere within an office building or campus, as well as from the corner café. In the home these LANs will enable a new class of intelligent home electronics that can interact with each other and with the Internet. Video teleconferencing will take place between buildings that are blocks or continents apart, and these conferences can include travelers as well, from the salesperson who missed his plane connection to the CEO off sailing in the Caribbean. Wireless video will be used to create remote classrooms, remote training facilities, and remote hospitals anywhere in the world.

The various applications described above are all components of the wireless vision. So what, exactly, is wireless communications? There are many different ways to segment this complex topic into different applications, systems, or coverage regions. Wireless applications include voice, Internet access, web browsing, paging and short messaging, subscriber information services, file transfer, and video teleconferencing. Systems include cellular telephone systems, wireless LANs, wide-area wireless data systems, and satellite systems. Coverage regions include in building, campus, city, regional, and global. The question of how best to characterize wireless communications along these various segments has resulted in considerable fragmentation in the industry,

as evidenced by the many different wireless products, standards, and services being offered or proposed. One reason for this fragmentation is that different wireless applications have different requirements. Voice systems have relatively low rate requirements (around 20 Kbps) and can tolerate a fairly high probability of bit error (bit error rates, or BERs, of around 10^{-3}), but the total delay must be less than 100 msec or it becomes noticeable to the end user. On the other hand, data systems typically require much higher data rates (1-100 Mbps) and very small BERs (the target BER is 10^{-8} and all bits received in error must be retransmitted) but do not have a fixed delay requirement. Real-time video systems have high data rate requirements coupled with the same delay constraints as voice systems, while paging and short messaging have very low data rate requirements and no delay constraints. These diverse requirements for different applications make it difficult to build one wireless system that can satisfy all these requirements simultaneously. Wired networks are moving towards integrating the diverse requirements of different systems using a single protocol (e.g. ATM or SONET). This integration requires that the most stringent requirements for all applications be met simultaneously. While this is possible on wired networks, with data rates on the order of Gbps and BERs on the order of 10^{-12} , it is not possible on wireless networks, which have much lower data rates and higher BERs. Therefore, at least in the near future, wireless systems will continue to be fragmented, with different protocols tailored to support the requirements of different applications.

3 Current State

Let us briefly summarize the current state of technologies on both wireless and video fields. On the wireless side, we have Digital Video Broadcasting (DVB) services (~ 20 Mbps) for broadcasting applications, Digital European Cordless Telephone (DECT) (~ 500 kbps) for short-range wireless communications, Global System for Mobile Communications (GSM) (~ 10 kbps) and in the near future 3G personal communication networks (~ 100 kbps).

On the video compression side we have HDTV format (1920x1080 pixels at 30 frames/sec) at ~20 Mbps, common intermediate format (CIF) (352x288 pixels ~ 10-20 frames/sec) at ~ 500 kbps and quarter common intermediate format (QCIF) (176x144 pixels ~ 10 frames/sec) at ~ 10 kbps. Although various data rates are possible, these correspond to DVB, DECT and GSM bandwidths, respectively.

4 Video Compression

In the last five years there have been extensive standardization efforts in digital video compression. The MPEG specification was developed specifically to allow the transmission of VCR-quality digital images at a data rate of approximately 1-1.5 Mbits/s. ISO1172 is also sometimes referred to as MPEG-1. Another specification known as MPEG-2 has been developed to support higher resolution images. Now after ISO's MPEG-1, MPEG-2, MPEG-4, and ITU's H-261 and H-263 standards and lots of proprietary algorithms, motion video compression can be considered a mature field. In this section we will briefly explain the basic schemes used in the standards mentioned above.

Video compression can be categorized into groups: pixel/frame based (e.g. MPEG-1, MPEG-2) and object based (e.g. MPEG-4). While the latter enables higher compression ratios for low and very-low bit-rate compression using specific models, the former is more commonly used. Hence, we will focus our attention on the former.

TABLE I
TYPICAL MPEG-1 AND MPEG-2 CODING PARAMETERS

| | MPEG-1 | MPEG-2 |
|-------------------------------|---|--|
| Standardized | 1992 | 1994 |
| Main Application | Digital video on CD-ROM | Digital TV (and HDTV) |
| Spatial Resolution | CIF Format (1/4 TV) appr. 288 x 360 pels | TV (4 x TV) appr. 576 x 720 pels (1152 x 1440 pels) |
| Temporal Resolution | 25 - 30 frames/s | 50-60 fields/s (100-120 fields/s) |
| Bit Rate | 1.5 Mbit/s | appr. 4 Mbit/s (appr. 20 Mbit/s) |
| Quality | comparable to VHS | comparable to NTSC/PAL for TV |
| Compression Ratio over PCM | appr. 20 - 30 | appr. 30-40 (appr. 30-40) |

The MPEG video compression algorithm relies on two basic techniques: block-based motion compensation for reduction in temporal redundancy and Discrete Cosine Transform (DCT) for reduction in spatial redundancy. MPEG is able to reduce the raw image data by twenty to fifty fold. Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined.

Intrapictures (I) pictures are coded without reference to other pictures and points for random access. I pictures use only intra-frame coding, based on the discrete cosine transform and entropy coding;

Predictive (P) pictures are coded with reference to previous I- or P-frames. P pictures use a similar coding algorithm to I pictures;

Bidirectionally predictive (B) pictures are coded with reference to a previous I- or P-frame as well as the future I- or P-frames.

The I picture frame at the beginning of a GOP serves as a basic entry point to facilitate random seek or channel switching and also provides coding robustness to transmission error, but it is coded with only moderate compression to reduce the spatial redundancies. P picture frames are coded more efficiently using motion compensated prediction from a past I or P picture frame and generally used as a reference for further prediction. B picture frames provide the highest degree of compression, but require both past and future reference pictures for motion compensation. It should be mentioned that B pictures are never used as references for prediction.

The structure of the GOP needs to be specified. This specification indicates the number of frames in a GOP, the number of P frames in a GOP, the number of B frames in a GOP, and the interleaving of the I, P, and B frames in the GOP.

The GOP structure is specified using two integer parameters, N and M . N specifies the number of frames in the GOP. M specifies the length of the substructure in

the GOP (i.e., after each I frame or P frame, there are $M - 1$ consecutive B frames before the next I or P frame). N must be an integer multiple of M .

Changing the GOP changes the average bit rate and burstiness of the MPEG sequence. For example, a GOP with mostly B frames will have a lower average bit rate than a GOP with just I frames and P frames. Note that simple desktop video conferencing applications (which often use just JPEG compression for each image) can be modeled with a GOP pattern of I (i.e., $N=1, M=1$), producing only I frames in the MPEG sequence. The default GOP pattern is IBBPBBPBB (i.e., $N=9, M=3$).

A GOP is defined by its length N_{GOP} , which is the distance between I pictures. As example of a GOP in MPEG is present in Fig. 2, where I, P, and B denote picture encoding in the intrapicture mode, predicted mode, and bidirectionally predicted mode, respectively. The GOP sequence of Fig. 2 with $N_{\text{GOP}} = 6$ is IBBPBBI. To maximize coding efficiency, both the motion compensation information and the transformed prediction error are represented using variable length (Huffman) codes (VLC).

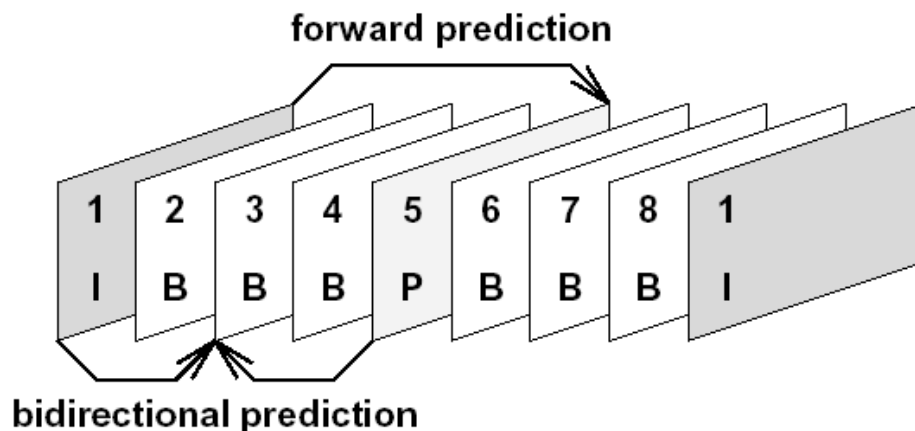


Fig. 2. Example of MPEG GOP

The macroblocks of 16 X 16 pixels are the basic coding units for the MPEG algorithm. Each macroblock is divided into four blocks, where each block contains 8 X 8 pixels. The main extension from monochrome video to color is the addition of two 8 X 8 chrominance blocks to the macroblock. A row of macroblocks that make up a horizontal strip in the image is called a slice and a number of slices are combined to form a picture. The coding mode of each macroblock within a specific picture depends on each block. The resulting two-dimensional block of DCT coefficients is quantized and scanned in a zig-zag order to convert it into a one-dimensional string of quantized coefficient data. The prediction pictures (P and B) use motion compensated prediction of the contents of the macroblock based on past or future reference pictures. This prediction is subtracted from the actual data in the current macroblock to form an error signal. The prediction error is coded like the intracoded macroblocks.

A simplified block diagram of this process is represented in figure 3.

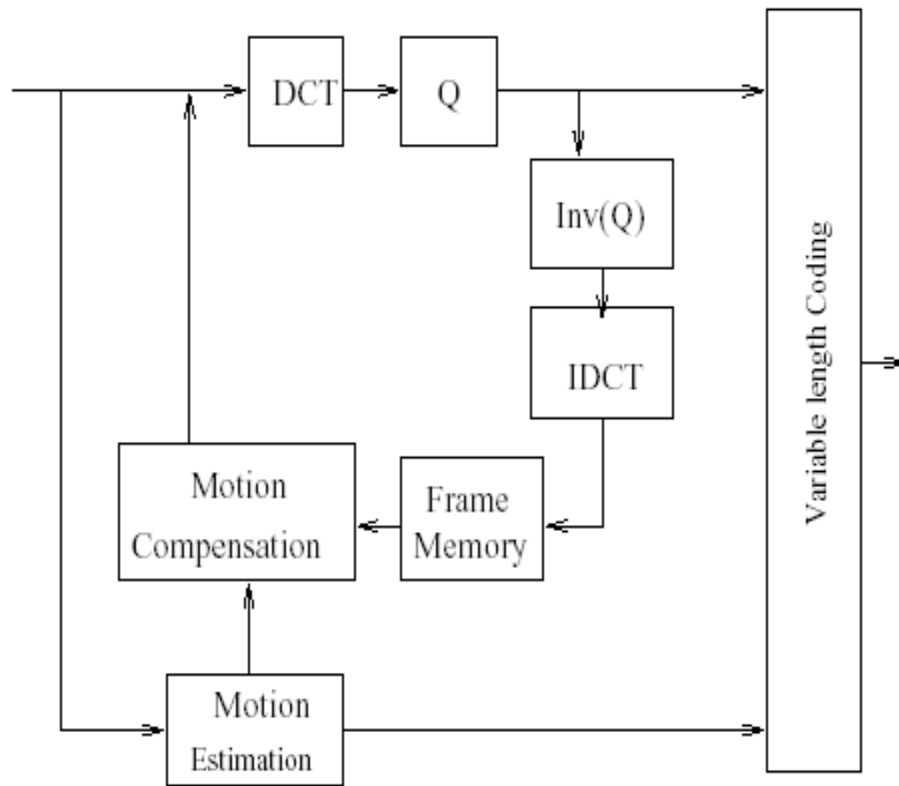


Figure 3: Block diagram for motion-compensated hybrid encoder

A couple of points in this scheme are especially important. The need to achieve high compression ratios in order to transmit digital video results in bit-streams that are highly sensitive to errors. This is a consequence of the relative information content of each bit increasing with the compression ratio. Moreover, an error in a bit-stream of variable length coded sequence is critical and results in loss of synchronization and huge errors. Also, differential coding of frames using motion compensation results in temporal propagation of errors. This propagation results in annoying effects for the viewer.

5 Wireless Channels

In a wireless mobile communication system, a signal can travel from the transmitter to the receiver over multiple reflective paths. This phenomenon is referred to as “multi-path” propagation. The effect can cause random fluctuations in the received signal’s amplitude and phase, known as multi-path fading. In second generation (2G) cellular systems (digital cellular), the effects of multi-path fading can typically result in as much as a 12-15 decibel (dB) drop in the received signal-to-noise ratio (SNR). Since SNR is directly related to the bit error rate (BER) of the receiver, multi-path fading results in large variations in the BER performance of the channel. In the other words, the error properties of a mobile channel are characterized as “bursty”. The frequency and length of the bursts are dependent upon the type of terrain (e.g., urban or mountainous) and the speed of the mobile unit.

Another important characteristic of wireless channels is low bandwidth. As the radio spectrum has to be shared by various applications, bandwidth is an extremely valuable resource and should be conserved whenever possible.

6 Solutions to Wireless Specific Problems

We will discuss a number of techniques, which require the participation of the sender of an audio stream to achieve recovery from packet loss. These techniques may be split into two major classes: active retransmission and passive channel coding. It is further possible to subdivide the set of channel coding techniques, with traditional forward error correction (FEC) and interleaving-based schemes being used. Forward error correction data may be either media independent, typically based upon exclusive-or operation, or media specific based on the properties of an audio signal.

In order to simplify the following discussion we distinguish a unit of data from a packet. A unit is an interval of audio data, as stored internally in an audio tool. A packet comprises one or more units, encapsulated for transmission over the network.

6.1 Forward Error Correction (FEC)

Error correction codes are widely used in context of communication systems. The basic principle behind the use of erasure code is that the original source data, in the form of a sequence of k packets, along with additional redundant packets, are transmitted by the sender, and the redundant data can be used to recover lost source data at the receivers. A receiver can reconstruct the original source data once it receives a sufficient number of packets. However, as the BER abruptly increases during fading in wireless channels, amount of redundancy that has to be added for the worst case is significant. Especially, in applications demanding high bandwidths, like video, this overhead becomes prohibitive. The main benefit of this approach is that different receivers can recover from different lost packets using the same redundant data. In principle, this idea can greatly reduce the

number of retransmissions, as a single retransmission of redundant data can potentially benefit many receivers simultaneously.

A number of forward error correction techniques have been developed to repair losses of data during transmission. These schemes rely on the addition of repair data to a stream, from which the contents of lost packets may be recovered. There are two classes of repair data, which may be added to a stream: those which are independent of the contents of that stream and those which use knowledge of the stream to improve the repair process.

6.1.1 Media Independent FEC

There has been much interest in the provision of media independent FEC using block, or algebraic, codes to produce additional packets for transmission to aid the correction of loss. Each code takes a codeword of k data packets and generates $n - k$ additional check packets for the transmission of n packets over the network.

A large number of block coding schemes exists and we discuss only two cases, parity coding and Reed-Solomon coding, as these are currently proposed as an RTP payload. These block coding schemes were originally designed for the detection and correction of errors within a stream of transmitted bits and so the check bits were generated from a stream of data bits. In packet streams we are concerned with the loss of entire packets and so we apply block coding schemes across the corresponding bits in blocks of packets. Hence the i 'th bit in a check packet is generated from the i 'th bit of each of the associated data packets.

Reed-Solomon codes are renowned for their excellent error correcting properties. Encoding is based upon the properties of polynomials over particular number bases. Essentially RS encoders take a set of codewords and use these as coefficients of a polynomial, $f(x)$. The transmitted codeword is determined by evaluating the polynomial for all non-zero values of x over the number base. Whilst this may sound complicated, the encoding procedure is relatively straightforward and optimized decoding procedures such as the Berlekamp-Massey algorithm are available. In the absence of packet losses decoding carries the same computational cost as encoding, but when losses occur it is significantly more expensive.

There are several advantages to forward error correction schemes. The first is that they are media independent: the operation of the forward error correction does not depend on the contents of the packets and the repair is an exact replacement for a lost packet. In addition, the computation required to derive the error correction packets is relatively small and simple to implement. The disadvantages of these schemes are the additional delay imposed, increased bandwidth and difficult decoder implementation.

6.1.2 Congestion Control

The addition of FEC repair data to a media stream is an effective means by which that stream may be protected against packet loss. However, application designers should be aware that the addition of large amounts of repair data when loss is detected will increase network congestion and hence packet loss, leading to a worsening of the problem which the use of FEC was intended to solve.

This is particularly important when sending to large multicast groups, since network heterogeneity causes different sets of receivers to observe widely varying loss rate: low-capacity regions of the network suffer congestion, whilst high-capacity regions are under-utilized.

So far, there is no standard solution to this problem. There are some suggestions. These typically use some form of layered encoding of data sent at different rate over multiple multicast groups, with receivers joining and leaving groups in response to long-term congestion and with FEC being employed to overcome short-term transient congestion.

Such a scheme pushes the burden of adaption from the sender of a stream to the receivers, which choose the number of layers (groups) they join based on the packet loss rate they observe. Since the different layers contain data sent at different rates, receivers will receive different quality of service depending on the number of layers they are able to join.

Layered encoding schemes are expected to provide a congestion control solution suitable for streaming audio applications. However, this work is not yet complete and it is important to give some advice to authors of streaming audio tools as to the behavior, which is acceptable, until such congestion control mechanisms can be deployed.

It has been suggested that one heuristic suitable for determining reasonable behavior for media streaming media tools is to adapt the transmission rate to the approximate throughput a TCP/IP stream would achieve over the same path. Since TCP/IP flows are the dominant form of traffic in the Internet, this would be roughly fair to existing traffic. Clearly such a scheme would not work for a multicast flow and clearly

it does not capture the dynamic behavior of the connection, merely the average behavior, but it does provide one definition of reasonable behavior in the absence of real congestion control. In the long-term, effective congestion control must be developed.

6.2 Interleaving

As mentioned above errors occur in bursts. FECs, designed for correcting random bit errors, become useless during bursts. By interleaving, errors during bursts are distributed among different blocks. Hence, FECs can correct these individual errors. However, interleaving introduces additional delays, i.e. more than one block has to arrive to start decoding. This may be critical in real-time video applications.



Figure 4: interleaving

6.3 Automatic Repeat ReQuest (ARQ)

ARQ techniques are generally used in unicast protocols: missing packets are retransmitted upon timeouts or explicit requests from the receiver. It is very simple and does not require any processing of the data stream. ARQ scales badly to multicast protocols and large groups, because the chance of uncorrelated losses grows with the size of the group, often requiring retransmission of every packet. This phenomenon is tolerable only in a few cases, e.g. when the group is small, receivers have similar features

and loss patterns, and they simultaneously start a transfer. In all other cases ARQ completely fails, since the aggregate packet loss rate can become exceedingly large. Another problem with ARQ is that it requires precise feedback from the receivers to decide which packets to retransmit; this leads to scalability problems and is unpractical in an environment where uplink communication is expensive.

Recently, it has been shown that truncated hybrid automatic repeat request (ARQ) schemes can significantly improve video transmission quality because of their adaptability to channel conditions.

There are two conventional classes of hybrid ARQ schemes: type-I and type-II. Type-I hybrid ARQ schemes include parity bits for both error detection and error correction in every transmitted packet. If the number of erroneous bits in a received packet is within the error correction capability of the code, the errors are corrected and the decoded message is accepted by the receiver. If an uncorrectable error pattern is detected, the packet is rejected and a retransmission is requested. The transmitter sends the original packet again. In type-I hybrid ARQ schemes, information can be recovered from each transmitted packet. Appropriately designed error correction overhead can correct the errors in the transmitted packets, thereby reducing the number of retransmissions compared to an error-detection-only pure ARQ scheme and enhancing the system throughput. Since wireless channels are time varying, some algorithms have been developed to adapt the code rate in each (re)transmission to the channel conditions. A disadvantage of type-I hybrid ARQ schemes is that the uncorrectable packets are discarded by the decoder even if they might contain some useful information.

For type-II hybrid ARQ schemes, the erroneous packet is kept for future use rather than discarded. Redundancy bits are transmitted only when they are needed, which provides the ability to adapt to changing channel conditions. In the conventional type-II hybrid ARQ schemes based on rate compatible punctured convolutional (RCPC) codes¹, the initial transmission of a data packet only includes the bits required for a very high-rate code. If the high-rate code is not powerful enough to combat channel errors, then supplemental bits, which were previously deleted by puncturing, are transmitted in the retransmission. The receiver, upon receiving the retransmitted redundancy bits, tries to recover the information by combining the redundancy bits with the previously transmitted data to produce a lower rate error correction code. If the combined decoding scheme fails, then this process continues as needed, decreasing the coding rate down to that of the mother code. If the starting code rate is high and the incremental step is small, it could give good throughput. A special case of type-II hybrid ARQ schemes, first introduced by Lin *et al.* [5], [6], employs a rate $\frac{1}{2}$ invertible block or convolutional code. The original data packet and the parity packet are alternatively transmitted until either an error-free packet is received or error correction is possible with the rate $\frac{1}{2}$ code obtained by combining the original data packet and the parity packet. However, each individual packet (information packet or parity packet) contains no error protection capability.

¹A low-rate $\frac{1}{2}$ code can be periodically punctured (i.e., deleting or not transmitting certain code bits) with period p to obtain a family of codes with rate $\frac{p}{p+b}$ where b can be varied between 1 and $(\frac{1}{2}-1)p$. The concept of RCPC codes was developed by Hagenauer [3]. For a family of RCPC codes, all code bits of high-rate codes can be used by the lower rate codes of the family; this allows transmission of incremental redundancy in hybrid ARQ schemes. We need to compromise the throughput to

For type-II hybrid ARQ schemes, the redundancy bits are added through retransmissions, that is, the type-II hybrid ARQ schemes trade off the number of retransmissions for a higher throughput. For real-time services, delay limits the number of retransmissions that can be used. It is impractical to increase the redundancy step by increase the error correction capability for faster convergence and to reduce the delay. Moreover, the conventional type-II hybrid ARQ scheme based on RCPC codes cannot recover information from the retransmitted packet alone [3], [4]. Both random and bursty errors exist on wireless channels. When a packet is in a long deep fade, almost all of the data in the packet are corrupted, and additional parity information for that packet will be useless for the decoder even if the parity retransmission is received with a high signal-to-noise ratio (SNR) value. In addition, the initially transmitted packet may be lost due to header errors so that the parity bits are completely useless. Although the type-II hybrid ARQ scheme based on rate $\frac{1}{2}$ block or convolutional codes can recover information from each transmission alone [5], [6], like a pure ARQ scheme, each transmitted packet itself contains no error correction capability. It is desirable that the information can be recovered from each transmission alone, and the transmitted packet itself has some error correction power dependent upon the channel conditions.

7 Tools for Digital Video Transmission on Wireless Channels

The solutions started above, which are applicable to voice and data transmission, are usually inefficient if not prohibitive for video transmission. Hence, specialized solutions should be devised. In this section, we will review these solutions. For the sake of analysis, we will categorize our solutions as encoder side, decoder side, and in the existence of reverse channel, as similar to many other transmission schemes, wireless channels can be analyzed in these parts as seen in figure 5.

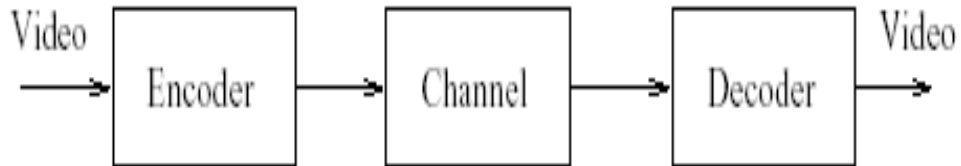


Figure 5: Block diagram of video over wireless channel

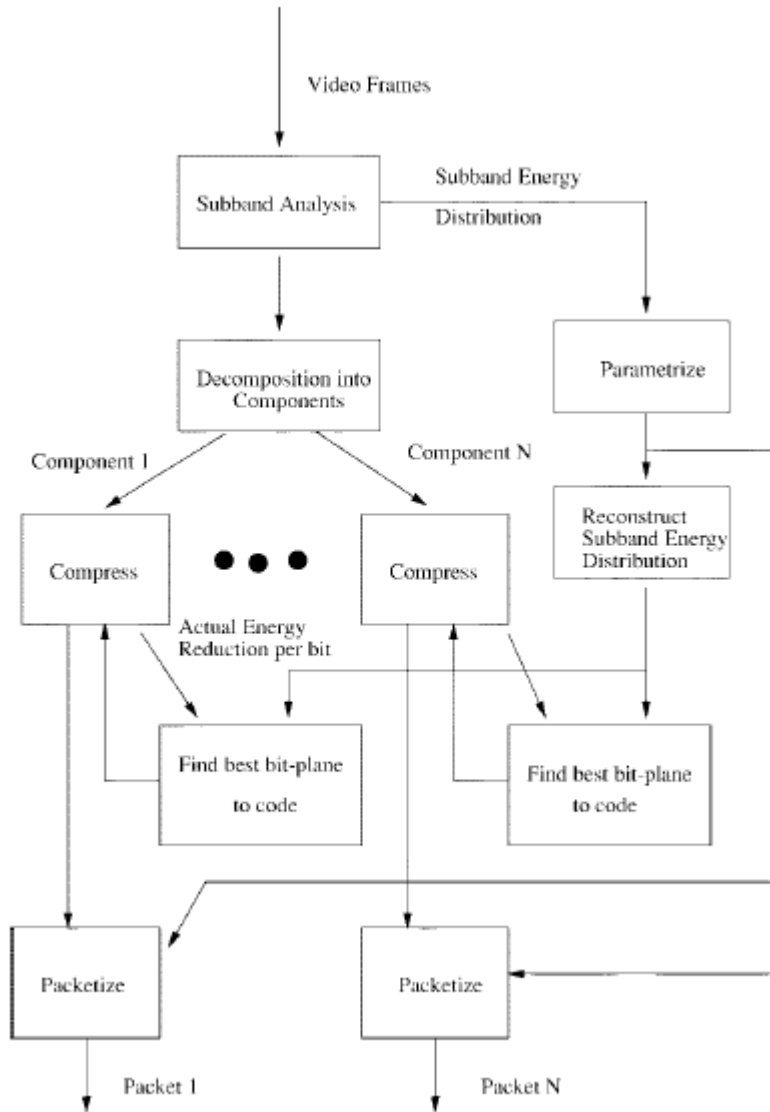


Figure 6: Encoding procedures.

7.1 Tools on Encoder Side

7.1.1 Resynchronization Markers

We mentioned earlier that VLC used in video compression is extremely vulnerable to bit errors and loose synchronization, resulting in total loss of data. In order to minimize this effort, video bit-stream is divided into packets of reasonable size and a resynchronization marker (a unique bit sequence) is inserted. In case of an error, only

data in one packet is lost and synchronization can be reestablished from the next resynch marker.



Figure 7: Resynchronization markers

A Video decoder decoding a corrupted bit stream, loses synchronization with the encoder (i.e. it is unable to identify the precise location in the image where the current data belongs). If remedial measures are not taken, the quality of the decoded video rapidly degrades and it becomes unusable. For this purpose, resynchronization markers (distinct from all possible VLC codewords) are introduced in the stream at various points (see **Fig. 8**) When the decoder detects an error, it can hunt for the resync. marker and regain resynchronization.

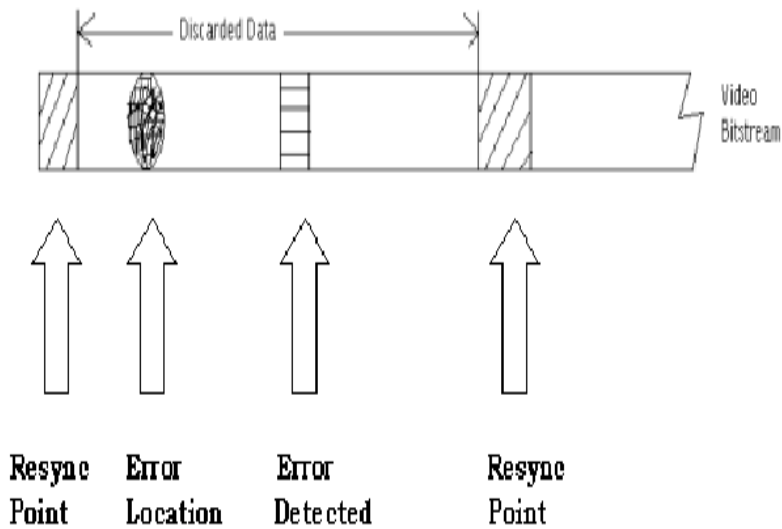


Fig. 8: Resynchronization markers present in bit stream

7.1.2 Concatenated FEC and Interleaving

A common method to add redundancy is forward error correction (FEC), which transmits redundant information of each packet in subsequent packets. In this sender based scheme, a lost packet can be recovered from the copies piggybacked in subsequent packets should they be received successfully. In this scheme, loss recovery is performed at the cost of higher latency [5]. In many cases, however, the loss of successive packets is correlated, due to the way packets are dropped as networks get congested and router buffers are becoming full. A packet loss may usually be followed by a burst of loss, which significantly decreases the efficiency of FEC schemes. In order to combat burst loss, redundant information has to be added into temporally distant packets, which introduces even higher delay. Hence, the repair capability of FEC is limited by the delay budget.

Another sender-based loss recovery technique, interleaving, which does not increase the data rate of transmission, also faces the same dilemma. The efficiency of loss recovery depends on over how many packets the source packet is interleaved and spread. Again, the wider the spread, the higher the introduced delay.

Although use of FEC codes and interleaving for worse case scenario is inefficient, they can be used to eliminate errors at lower BERs. The best results are obtained by concatenation of convolutional and block error correction codes. Block diagram for such a system is shown in figure 9.

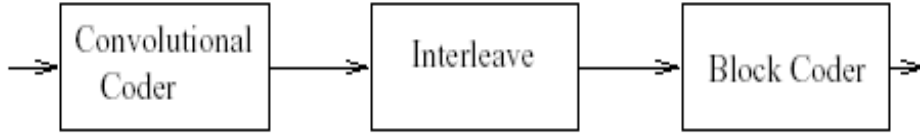


Figure 9: Concatenated FEC

7.1.3 Data Partitioning and Unequal Error Protection (UEP)

Although video compression algorithms enable high compression rates they are far from perfect. Information content, hence importance and motion vectors are more vulnerable to errors. Data partitioning is grouping the bit w.r.t. their relative importance. After data partitioning, unequal error protection can be used to heavily protect more important parts of the bit-stream.

RCPC codes [8] are constructed by puncturing a convolutional code called the parent code. Let the code rate and constraint length of the parent code be $R = 1/n$ and L_C , respectively. The parent code is completely specified by the n generator polynomials $G^j(D) = g^j_0 + g^j_1 D + \dots + g^j_{L_C-1} D^{L_C-1}$, $j = 1, 2, \dots, n$, where $g^j_i \in \{0, 1\}$ [12]. The puncturing is done according to the rate compatibility criterion, which requires that lower rate codes use the same coded bits as the higher rate codes plus one or more additional bit(s). The bits to be punctured are described by an $n \times p$ puncturing matrix P consisting of zeros and ones, where p is called the puncturing period. At time instant t , the output from each generator $G^j(D)$ is transmitted if $P(j, t \bmod p) = 1$ and punctured otherwise. Here, $P(a, b)$ denotes the element on row a and column b in the matrix P . The number of columns determines the number of code rates and the rate resolution that can be obtained.

Generally, from a parent code of rate $1/n$, we can obtain a family of $(n - 1)p$ different codes with rates $r = p/np, p/np - 1, \dots, p/p+1$ (8)

The code rate of RCPC codes can be changed during even one information bit transmission and, thus, unequal error protection can be obtained [8]. In this paper, however, the code rate of RCPC codes is changed frame by frame, not bit by bit, because we assumed frame by frame transmission. An example of the RCPC encoder is shown in Fig. 6.

Rate Compatible Punctured Convolutional(RCPC) codes are especially useful UEP. In RCPC codes, data is first coded using a low rate convolutional coder. Higher rates are simply obtained by puncturing, i.e. not transmitting certain locations of the coded bit-stream. Decoder simply inserts zero in places of punctured bits and decodes the bit-stream.

7.1.4 Reversible Variable Length Codes (RVLC)

RVLCs are prefix codes that admit decoding in either direction. For example, a decoder can begin by processing the received bit stream in the forward direction, and upon detecting an error, proceed immediately to the end of the bit stream and decode in the reverse direction. This, and related strategies can be used to significantly reduce the effects of bit errors on the fidelity of the decoded data.

In contrast with the enormous amount of work that has been performed on general VLCs, there has been relatively little formal study of issues specific to RVLCs. One of

the first publications to treat RVLCs was by Takishima, Wada and Murakami, who studied the conditions for the existence of RVLCs and proposed algorithms for their construction. More recently, two sets of RVLC tables were proposed for inclusion in MPEG-4 by Toshiba and Ericsson respectively, and after testing in a core experiment, the RVLCs proposed by Toshiba were accepted for coding of DCT data. The ITU document included as part of Annex D of H.263+.

In a VLC bit-stream, in case of an error, data till the next resynchronization marker has to be discarded. However, data between the error point and resynchronization marker is error free. In order to recover this portion of data MPEG-4 utilizes reversible variable length codes (RVLC). This type of coding, despite some loss of efficiency, can be decoded starting from the end of the bit-stream. Hence, in case of an error as shown in figure , bit stream can be decoded first from resynch marker to error, than reversely decoded from next marker to the error. Despite the increased complexity and overhead size, this method has shown to significantly improve performance.

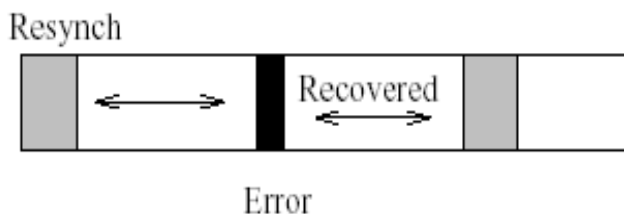


Figure 10: Reversible Variable Length Codes

7.1.5 Joint Source-Channel Coding

Although the joint source-channel coding theorem of Shannon [2] implies that source coding and channel coding can be treated separately without any loss of overall

performance, this separability holds only if the communication is point-to-point and allows infinite codeword length, which is not realistic in a practical environment with finite delay or a channel with multi-path fading. Joint source-channel matching takes advantage of this unsatisfied prerequisite to achieve performance gain and a certain level of error resilience.

Because the separation principle is based on arguments that hold only in the limit, a channel-coding scheme converges to the theoretic optimum only if we process an arbitrarily large amount of data and are willing to incur unbounded coding delays. In other words, separation assumes that we do not impose a delay constraint at the source. For H.320, large encoding delays are avoidable because transmission errors in the underlying ISDN circuit occur at a bit-level granularity (not typically in bursts) and can be corrected by simple and short block error codes.

On the other hand, separation does not necessarily apply to other typical communication environments. Cover showed that, for a certain model of broadcast channel, performance can be improved by superimposing the delivery of low-rate and high-rate information [13], e.g., by distributing video to heterogeneous receivers in a layered format. Hence, performance can be improved by modifying the source-coding algorithm (i.e., layered compression) to better match the channel-coding algorithm (i.e., layered transmission).

In other words, we achieve improved performance through joint source/channel coding, and clearly, our proposed layered video architecture is an instance of JSCC. Source and channel coding can be separately designed. If both the systems are perfectly designed, separation is not important. However, due to time and complexity constraints

both source and channel coding are far from being perfect. Basically, there is still some redundancy in the compressed video and some errors can still occur after channel coding. Joint source-channel coding schemes explore this redundancy in the compressed video, called residual redundancy. They also incorporate the channel characteristics to employ more effective channel coding.

The advantages of joint source-channel coding for image transmission have been extensively studied; here we present an incomplete list of previous research: Davis and Danskin [3] described a joint source-channel allocation scheme for transmitting images losslessly over block erasure channels such as the Internet; Ramchandran *et al.* [4] studied multiresolution coding and transmission in a broadcasting scenario; Azami *et al.* [5] acquired performance bounds for joint source-channel coding of uniform memoryless sources using binary decomposition; Belzer *et al.* [6] developed a joint source-channel image coding method using trelliscoded quantization and convolutional codes; Sherwood and Zeger [7] investigated unequal error protection for the binary symmetric channel; Man *et al.* [8] examined unequal error protection and source quantization; Lu, Nosratinia and Aazhang developed a closed-form solution for progressive source-channel coding of images over bursty error channels [9]; Fossorier, Xiong and Zeger studied joint source-channel image coding for a power constrained noisy channel [10]. A general method which can be applied to most source and channel coders can be found in [11].

Joint source-channel matching for wireless video transmission, on the other hand, is much less mature because of the extra dimension and other new challenges. Bystrom and Modestino investigated combined source-channel coding for video transmission over a slowfading Rician channel [12], Lan and Tewfik studied power-optimized mode

selection for H.263 video coding in wireless communication [13], Zheng and Liu used a subband modulation approach to transmit image and video over wireless channel [14].

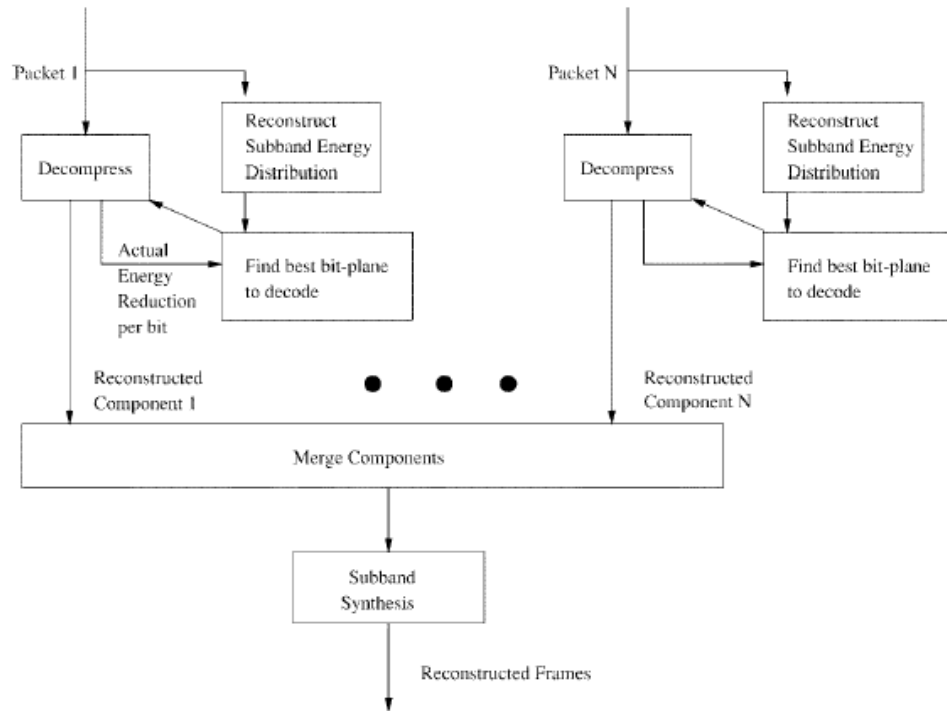


Figure 11: Decoding procedures

7.2 Tools on Decoder Side

Having completed the encoder side, now let us consider the methods that can be used on the decoder side.

7.2.1 Error Detection

The most important thing on the decoder side is the detection of errors. Without detection, effective counter-measures can not be taken. Error detection can be performed in two levels. In the first level, error detection properties of FEC codes are used. On the second level, semantic checks can be employed. Semantic checks can vary in complexity.

Two such checks are: (i) number of coefficients from an 8x8 block should be 64 (ii) each code in VLC stream should be in the codebook. Using FECs and semantic checks, good error detection and localization is provided.

For example, Java based Extraction and Dissemination of Information (Jedi) is one of tools performs semantic checks. For performing semantic checks on the syntactically matched data, Jedi supports predicates that can, for example, check for number ranges or lookup a thesaurus. Predicates can be defined by embedding arbitrary rule productions between 'accept' and 'if' keywords. Whenever the embedded production matches the source, the predicate expression following the 'if' is checked against the matched data passed as a special parameter '\$\$'. If the predicate expression evaluates to true, the production match is accepted, otherwise it is rejected.

The following example illustrates the use of predicates to disambiguate between street numbers and zip codes of addresses. Because both are defined by equivalent structural descriptions ([0-9]+), and both may be missing, they can neither be disambiguated by the local pattern nor by the syntactic sequence. Thus we need predicates that constrain street numbers to a length smaller than 5 and (German) zip codes to be of length 5:

rule address is

// ... - some rules omitted for simplicity

streetname()

(

accept

```

[0-9]+ // pattern for street numbers
if $$length<5;
)?)?
.* // skip non relevant characters
(
accept
[0-9]+ // same pattern for zipcode
if $$length == 5;
)?)?
.* // skip non relevant characters
cityname()
end

```

7.2.2 Soft In/ Soft Out Decoders

The soft-input soft-output (SISO) module, an algorithm which is similar to the Viterbi Algorithm (VA), accepts *a priori* information on the input and output symbols of a finite state machine (FSM) and outputs the corresponding *a posteriori* information, with complexity growing linearly with the record length.

The objective of a SISO algorithm is to provide soft information about the input and output symbols of the FSM based on the observation record. This reliability information can either be in the form of an *a posteriori* probability or any other related quantity. It would be advantageous at this point to generalize the notion of the state s_k and

transition t_k to longer sequence portions (e.g., a super-state and super-transition can be defined as $s_k^s = (t_{k-d}, \dots, t_{k-1}, s_k)$ and $t_k^s = (t_{k-d}, \dots, t_{k-1}, t_k)$ for arbitrary). This foreshadows the result that the optimal algorithms do not “fold” [22] onto a trellis as in the case of known channel and that the size of the trellis eventually used is a design parameter. For a generic quantity u_k (i.e., $x_k, y_k, s_k, t_k, s_k^s, t_k^s, \dots$), we define the *a posteriori* probability (APP) and minimum sequence metric (MSM) soft outputs as follows:

$$\text{APP}_p(u_k) = P(u_k | z_0^n, x_0^n) = c \sum P(z_0^n, x_0^n) = c \sum E_\theta \{ P(z_0^n, x_0^n | \theta) \} \quad (9a)$$

$$\text{MSM}_p(u_k) = -\log [\max_{x_0^n} P(z_0^n | x_0^n)] = c' -\log [\max_{x_0^n} P(z_0^n, x_0^n)] = c' -\log [\max_{x_0^n} E_\theta \{ P(z_0^n, x_0^n | \theta) \}] \quad (9b)$$

where $x_0^n : u_k$ denotes all input sequences consistent with u_k , and c, c' and are normalizing constants. These soft outputs are the direct generalizations of well-known soft outputs for perfect CSI [7] to the case of an unknown parameter θ . When the SISO module is part of an iterative receiver, the soft output is usually normalized to the *a priori* information resulting in the so-called extrinsic information (e.g., $\text{APP}_p(u_k) / P(u_k)$, or $\text{MSM}_p(u_k) - (-\log P(u_k))$ is used in place of $\text{APP}_p(\cdot)$ or $\text{MSM}_p(\cdot)$, respectively). We observe that in both cases, the soft outputs can be derived from the quantity $E_\theta \{ P(z_0^n, x_0^n | \theta) \}$ by either averaging or maximizing—for $\text{APP}_p(\cdot)$ or $\text{MSM}_p(\cdot)$, respectively—over the nuisance parameters $x_0^n : u_k$.

Equation (9) clearly suggests a way of manipulating $P(z_0^n, x_0^n | \theta)$ to obtain the proposed soft metrics. Maintaining the conditioning over the entire input sequence, expectation can be performed on the unknown parameter. Combining of the resulting metrics over the nuisance parameters $x_0^n : u_k$ is performed as a final step, leading to the final two soft metrics for u_k . Since operators $\sum_{x_0^n : u_k}$ and E_θ commute, an additional

choice is available for the evaluation of the metric in (9a). Here, the sequence combining is done initially, followed by the parameter elimination. Different soft metrics can also be defined by interchanging the operator $\max_{x^n_0 : u_k}$ with the operator E_θ in (9b). This option will not be pursued in this work, mainly because it does not appear to lead to rigorously expressed optimal structures.

Soft input/output decoders like turbo-codes are quite popular in wireless applications. In soft input case, instead of assuming equal probabilities for each code/bit, a priori probabilities extracted from source decoder can be used. In soft output case, instead of making hard decisions a posteriori probabilities of each code/bit can be provided. Possible errors can be corrected in source decoder using relative probability of received bits. Viterbi and MAP decoders are especially suitable for these applications.

7.2.3 Temporal Extrapolation

As the error correction methods fail occasionally, tools that minimize the effects of errors should be used. Temporal extrapolation is one of them. If a group of pixels are received with error, those pixels can be estimated using previous frames. As the temporal redundancy increases, this method can be quite powerful.

Temporal interpolation replaces the missing region in a frame with the corresponding region in a previous frame. If motion vectors are available, temporal interpolation replaces the missing region with the region specified by the vectors.

The first motion vector of the lost block is obtained using previous frames. Afterwards, a motion compensated block replaces the lost block. However, motion

estimation is quite complex. So, usually motion vectors are assumed to be zero and the same group of pixels from the previous frame is used.

The disadvantages of conventional image interpolation algorithms can clearly be illustrated by applying a six degree of freedom rigid body transformation to a test image, then applying the inverse transformation to return the image to its original location, and finally subtracting the test image from the twice-transformed image.

Sampling theory provides part of the solution. The k-space image representation of a correctly sampled band limited function can be recovered from the discrete Fourier transform by multiplication with a cubical window. The corresponding spatial domain operation is a convolution with the Fourier transform of the cubical window, which is a 3D subband function. Image transformations incorporating rotations are better carried out in the spatial domain rather than in k-space, requiring the use of a subband as the interpolation function.

Real images are truncated in all three dimensions, so are not band-limited. Consequently subband interpolation alone does not correctly transform the images. The most striking artefact introduced by this is shading in the rotated image plane. In addition, the subband interpolation function is of necessity truncated to make it possible to transform images in a reasonable length of time.

The truncation problem cannot be entirely eliminated, but can be reduced by image extrapolation and windowing the subband function. We extrapolate the image using a reflected copy of itself in directions in which the subject is truncated, and a duplicate copy of itself elsewhere (the latter is computationally cheaper).

7.2.4 Spatial Interpolation

Spatial interpolation is another way of approximating the data with errors. There are mainly two methods of spatial interpolation: in spatial domain and in frequency domain.

Spatial interpolation reconstructs a missing piece in a frame from its adjacent (presumably non missing) regions in the same frame. Its performance is mediocre for block transform coders such as the H.261 coder used in IVS because several consecutive lines are damaged when a packet is lost.

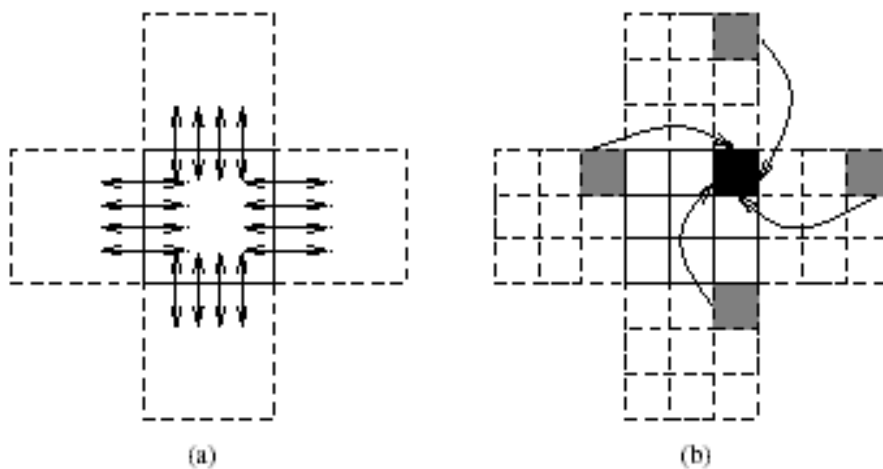


Figure 12: Spatial interpolation of lost pixels (a) Spatial domain (b) DCT domain

7.2.5 Motion Vector Interpolation

In case of a lost motion vector for a block, two methods similar to pixel recovery can be used. It may be obtained extrapolating the motion vectors from previous frames, like temporal extrapolation, or it may be recovered using smoothness constraints and spatial interpolation. Note that these methods provide good results only when certain temporal and spatial smoothness constraints are satisfied.

Zonal based algorithms can achieve a smoother motion vector field, and relatively more accurate, and closer to the true motion, prediction of the motion vectors compared to other algorithms, including FS.

This can be also observed from Table 2 where the entropy of the difference of the current motion vector versus other closely related motion vectors of spatially and temporally adjacent blocks is given. To understand these numbers, consider that adjacent blocks are highly correlated, and thus, in reality, tend to have also similar motion vectors. This would also suggest that, in most cases, the closer to the true motion field an algorithm gives, the smaller the entropy versus a set of possible predictions would be as well. These predictors include the motion vectors of the three spatially adjacent blocks on the left, top, and top right to the current position, their Median, the (0,0) motion vector, and even the motion vector of the collocated block in the previous frame. All these predictors constitute a Predictor Set (Set A), and are quite important in the performance of zonal algorithms. Obviously as we can see in Table 2 zonal algorithms have significantly lower entropy compared to other algorithms such as FS, and combined with Figure 12 we may claim that they give motion vector field that is rather reliable and close to the true motion.

For the purpose of our experiments we have selected using the Predictive Motion Vector Field Adaptive Search Technique (PMVFAST). This algorithm can be essentially considered as a special case of the Zonal Algorithms. Other zonal algorithms such as the Advanced Predictive Diamond Zonal Search (APDZS) [12] could also be selected with similar or better performance. It should be noted that PMVFAST was recently accepted as a recommendation for motion estimation in the Optimization Model 1.0 of MPEG-4

[13] and it is characterized by its superior speed up and quality versus most if not all other algorithms. This is quite important in a practical implementation since, considering that for a specific sequence motion vectors could possibly have been already estimated using PMVFAST, we may immediately use these motion vectors for deinterlacing, without having to repeat the entire motion estimation process.

| | Full Search | | | Zonal Algorithm | | |
|----------|-------------|--------|-------|-----------------|--------|-------|
| Sequence | (0,0) | Median | Set A | (0,0) | Median | Set A |
| Foreman | 6.78 | 5.17 | 4.64 | 6.70 | 4.11 | 2.50 |
| Stefan | 7.07 | 4.96 | 4.41 | 7.05 | 4.03 | 2.07 |
| Bus | 6.86 | 5.31 | 4.37 | 5.89 | 3.36 | 1.51 |

Table 2: Motion Vector Entropy versus different Predictors

We first perform field motion estimation, using a block size of 16×8 , using the fields of interest (i.e. the even fields) from the current and previous frames. By doing so we may generate the motion field for a time interval t . What interests us is to make an estimate of the motion field at a time interval $t/2$. This is achieved by simply dividing all motion vectors by 2. Afterwards using motion compensation techniques we can generate the missing field lines. These missing fields may be then interleaved with the original fields at time interval $t/2$ (in this case the odd fields) in order to generate the progressive frame.

Motion estimation using non-corresponding fields could also be used to further improve performance.

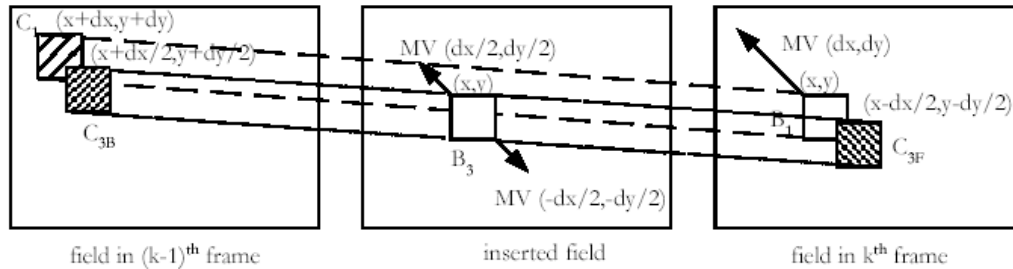


Figure 13: A simple motion vector interpolation method for predicting the missing field

One question that may arise is what is the best method to perform motion compensation. Let us assume that we have performed motion estimation for the corresponding fields in the k th frame versus the $(k-1)$ th frame. Let us assume that block B_1 at position (x, y) has a motion vector (dx, dy) . The easiest method for constructing the missing field, but nevertheless a not so accurate one, would be to assume that the block at the corresponding (x, y) position inside the missing field B_3 also has a motion vector equal this time to $(dx/2, dy/2)$ pointing in the $(k-1)$ th frame (block C_{3B}). This can be seen in Figure 3. We may also assume that this same block has a second motion vector, this time equal to $(-dx/2, -dy/2)$ which this time points to the k th frame (block C_{3F}). Thus B_3 can be reconstructed as the average of blocks C_{3B} and C_{3F} . This is rather similar to the way B frames are reconstructed, and its advantage is its relative simplicity, especially for hardware, since we may reuse the standard motion compensation architecture.

7.3 Tools in the Existence of Reverse Channel

Till now we assumed that there is only a one-way link between the transmitter and receiver, which is the case in broadcasting applications. However, a reverse link from receiver to the transmitter can significantly improve performance. In this section we review these methods.

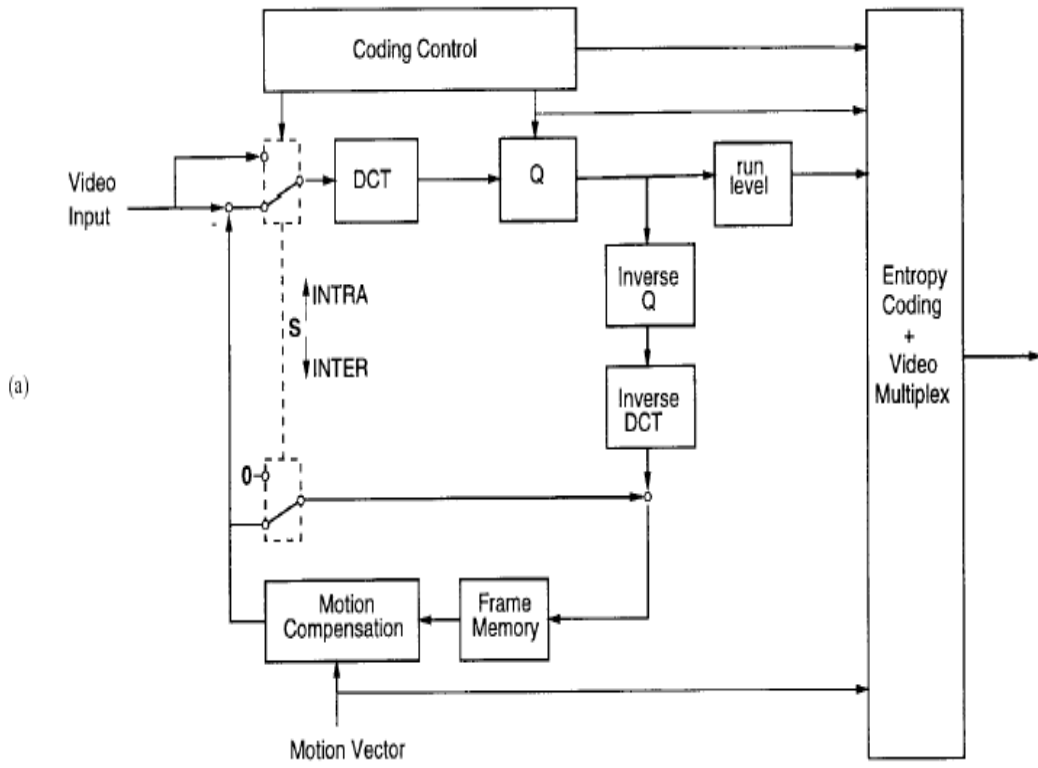
7.3.1 Automatic Repeat ReQuest (ARQ)

ARQ is one of the methods mentioned earlier as a general solution. ARQ mechanisms are closed-loop mechanisms based on the retransmission of the packets that were not received at the destination. Location of the error is fed back to the transmitter and retransmission of packet with error is requested. Note that this implies a round-trip delay of typically ~ 200 ms, which is prohibitive for real-time point-to-point communication. Another problem is the increase in required bandwidth, which is used to retransmit the packet. Problems make ARQ a very inefficient solution to the problem, and so it is not commonly used. ARQ mechanisms are typically not acceptable for live audio applications over the Internet because they dramatically increase end to end latency.

7.3.2 Error Tracking

The error tracking approach uses the INTRA mode for some macroblocks (MB's) to stop interframe error propagation but limits its use to severely affected image regions only. During error-free transmission, the more effective INTER mode is used and the system therefore adapts effectively to varying channel conditions. This is accomplished by processing the negative acknowledgement (NACK's) from a feedback channel in the

coding control of the encoder (Fig. 14). Based on the information of an NACK, the encoder can reconstruct the resulting error distribution in the current frame as described below. The coding control of a forward-adaptive encoder can then effectively stop interframe error propagation by selecting the INTRA mode whenever a MB is severely distorted. On the other hand, if error concealment was successful and the error of a certain MB is small, the encoder may decide that INTRA coding is not necessary. For severe errors however, a large number of MB's is selected, and the encoder may have to use a coarser quantizer to maintain a constant frame rate and bit rate. In this case, the overall picture quality decreases with a higher frequency of NACK's. Unlike retransmission techniques such as ARQ, error tracking does not increase the delay between encoder and decoder. It is therefore particularly suitable for applications that require a short latency.



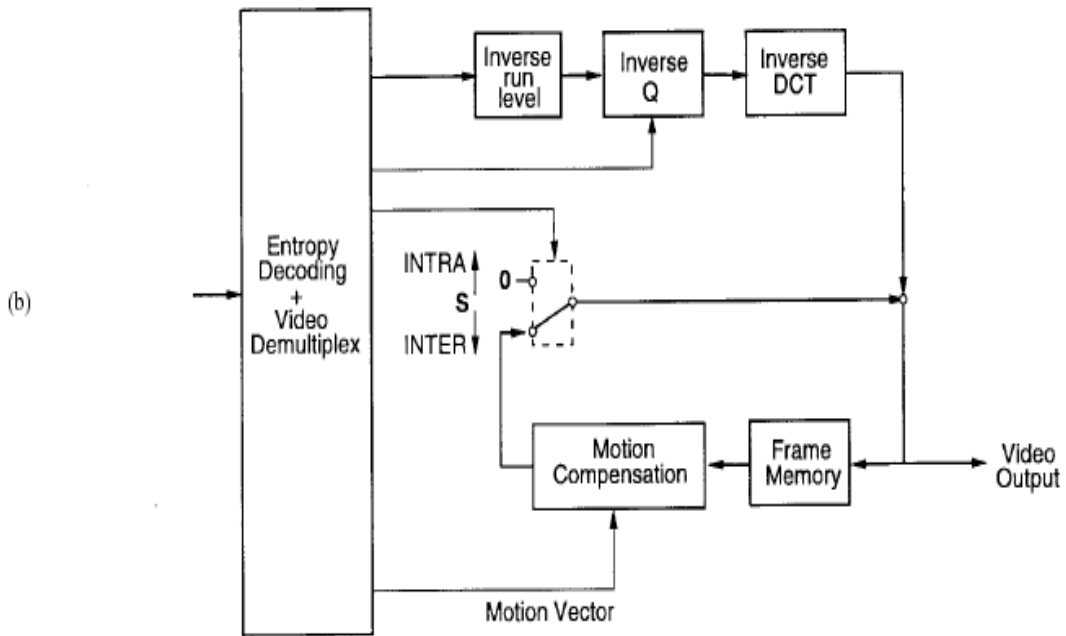


Figure 14: (a) Motion-compensated hybrid encoder and (b) decoder

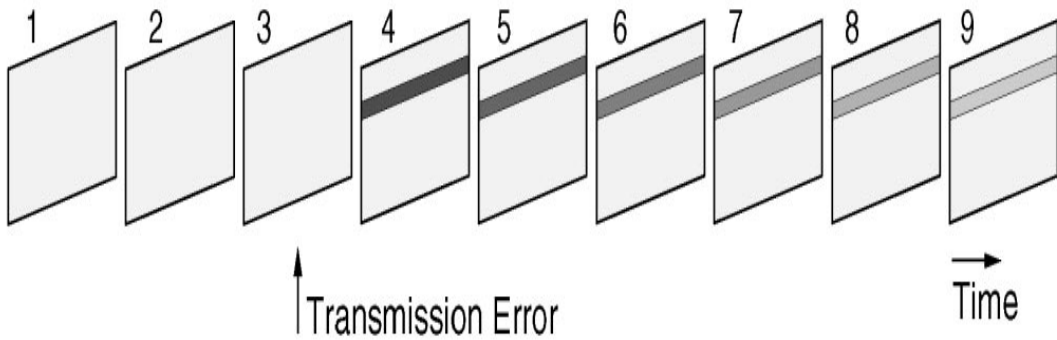


Figure 15: Illustration of spatiotemporal error propagation

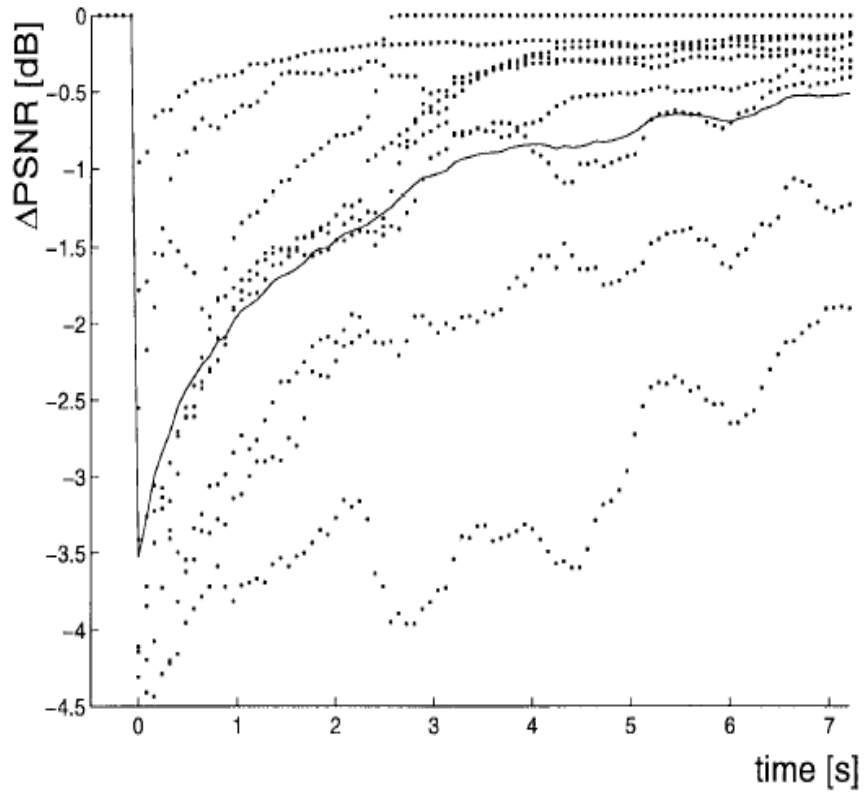


Figure 16: Loss in SNR of the decoded video signal after previous frame concealment of one GOB

Fig. 16 illustrates error tracking for the same example as in Fig. 15. As soon as the NACK is received with a system-dependent round-trip delay, the impaired MB's are determined and error propagation can be terminated by INTRA coding these MB's (frames 7–9). Fig. 17 shows the averaged PSNR loss for an assumed round-trip delay

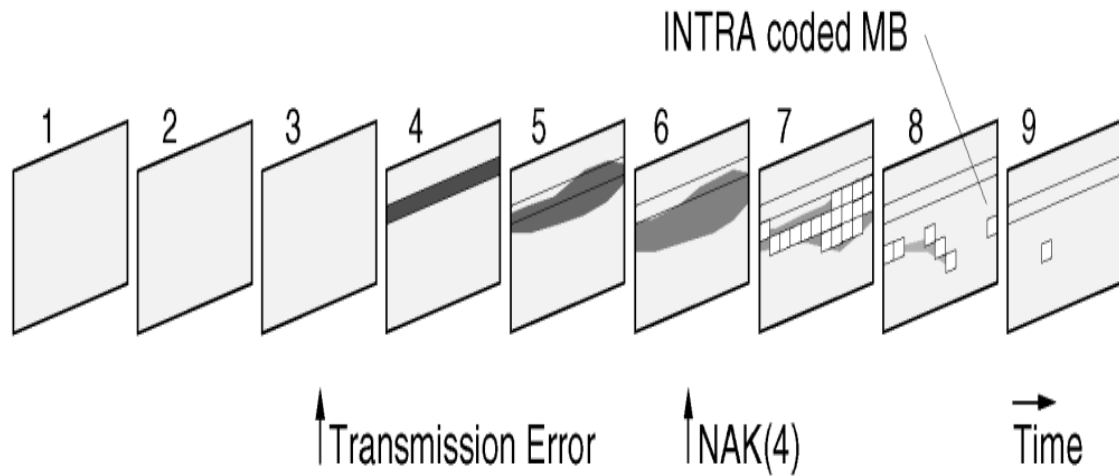


Fig. 17. Illustration of error propagation when error tracking is used.

of 800 ms. The same simulation conditions as in Fig. 15 are used. Compared to the case without error tracking, the picture quality recovers rapidly as soon as INTRA coded MB's are inserted into the bit stream. A longer round-trip delay just results in a later start of the error recovery. Considering the slow recovery for concealment only, NACK's may still be useful after several seconds. In order to illustrate the importance of actually tracking the shifting location of the error, we also show results for the simple same GOB strategy, where the error is assumed to remain in the GOB were it originally occurred. When errors are dragged out of the original GOB along with vertical motion of picture contents, the same GOB strategy cannot remove it successfully, and annoying artifacts remain. Only when error tracking is employed, the propagation is stopped effectively. This is also demonstrated in Fig. 17, which shows example frames of the H.263 encoded test sequence *Foreman* directly after the NACK is received.

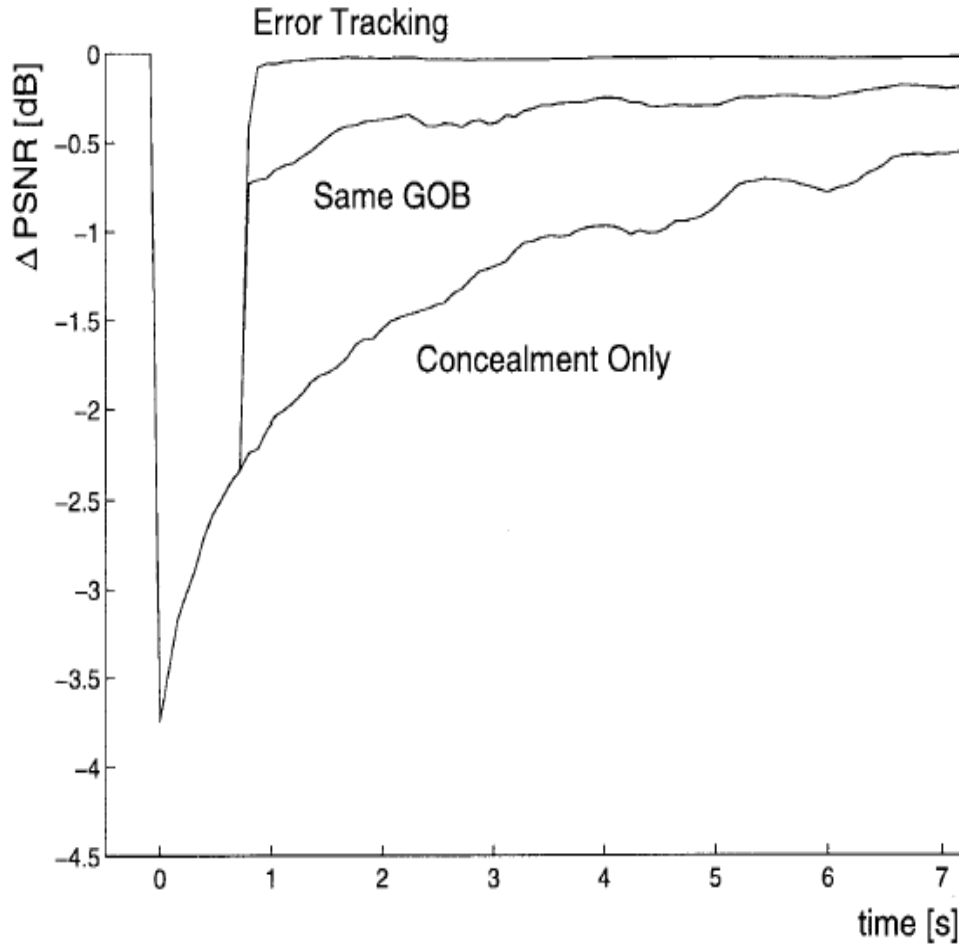


Fig. 18. Error recovery with feedback channel.

In order to reconstruct the interframe error propagation that has occurred at the decoder, the encoder could store its own output bit stream and decode it again, taking into account the reported loss of GOB's. While this approach is not feasible for a real-time implementation, it illustrates that the encoder, in principle, possesses all the information necessary to reconstruct the spatiotemporal error propagation at the decoder, once the NACK's have been received. For a practical system, the interframe error propagation has to be estimated with a low-complexity algorithm, as described in [17]–[18] for a macroblock-based coder, such as H.263.

The basic idea of the low-complexity algorithm is to carry out the error tracking with macroblock resolution rather than pixel resolution. This is sufficient since the INTRA/INTER mode decision at the coder and the error concealment decision at the decoder are carried out for entire MB's as well. In a cyclical buffer that covers all the MB's of the last several frames, the spatial overlap of MB's in successive frames due to motion-compensated prediction is stored, along with the error energy that would be introduced if concealment had to be used. If an NACK is received that indicates an error a few frames back, this error energy is "released" and "ripples" through the directed graph of frame-to-frame dependencies to the macroblocks of the current frame. The interframe error propagation model derived in the Appendix can be incorporated for more accurate prediction. Since all calculations are carried out on the MB level, the computational burden and memory requirements are small compared to the actual encoding of the video. For example, at QCIF resolution, there are only 99 MB's in each frame, as opposed to 38 016 luminance and chrominance samples.

The above error tracking scheme is a refinement of Wada's "selective recovery method" [15]. Wada's method marks all potentially damaged image blocks by one bit and prevents their use for interframe prediction. The method described here also calculates the severeness of the impairment and can thus use the extra bits required for INTRA coding more sparingly. Note that the coder has to know the decoder's concealment technique for that. On the other hand, Wada's method only considers the worst possible interframe error propagation and hence does not require an agreed concealment technique.



Fig. 19. Reconstructed frames of test sequence *Foreman*: (a) frame 90 after previous frame concealment of two GOB's in frame 75; (b) as (a) with INTRA update in frame 90 according to Same GOB strategy; (c) as (a) with INTRA update in frame 90 according to error tracking strategy; (d) frame 90 without GOB loss in frame 75.

Error tracking is particularly attractive since it does not require any modifications of the bit stream syntax of the motion-compensated hybrid coder. It is therefore fully compatible with standards such as H.261, H.263, or MPEG. The ITU-T recommends using previous frame concealment and error tracking with baseline H.263 and has included an informative appendix with Recommendation H.263. In addition, minor

extensions of the H.245 control standard were adopted to include the appropriate NACK messages.

As in ARQ case, in error tracking location of the error is fed back to the transmitter. However this time, instead of retransmitting the packet, transmitter simulates the possible result of the error and codes next frames to stop propagation of error. A very simple example is leaving motion compensated coding mode and coding the frame independent of previous frames. Although the frame with error is shown on decoder side, temporal propagation is prevented. Due to temporal masking effects of human visual system annoying effects are not seen by the viewer. Note that intra-frame coding is less efficient than motion compensated coding. Thus this scheme results in increased bandwidth as well.

More complicated error tracking and compensation techniques exist. They usually offer lower increases in bandwidth with increasing complexity.

7.3.3 Joint Source-Channel Coding

Joint source-channel coding has been mentioned earlier as a method that can be used on encoder side. In that case, we only assumed that encoder knows the a priori characteristics of the channel, and optimizes source and channel coding parameters in return.

Since wireless channels' characteristics vary significantly in time, joint source-channel coding can be improved if time-varying characteristics of the channel are known. Basically, the reverse channel can be used to feedback the channel state to the transmitter, e.g. bit error rate.

Depending on the channel state feedback, encoder can allocate bandwidth between source and channel coding. For example, in case of high BER, a lower resolution version of video is more strongly error correction coded.

Scalable coding techniques are optimized to deliver maximum quality video for a given bandwidth. In case the encoded bit stream is less than the specified bit-rate, bandwidth is wasted and if the bit stream exceeds the available bit-rate, large amount of coded information is lost. All Internet applications, where bandwidth changes dynamically, require coding schemes that can give maximum quality at any given bit-rate. This is the main objective of FGS video coding.

FGS uses bit-plane coding of Discrete Cosine Transform (DCT) coefficients of the residual frame. A bit-plane of the block is defined as an array of 64 bits, taken one from magnitude of each of the DCT coefficients at the same significant position, scanned in zigzag order. The maximum number of bit-planes coded in a frame, N , is given by

$$N = \log_2 (|\text{DCT}|_{\max}) + 1 \quad (1)$$

where $|\text{DCT}|_{\max}$ is the magnitude of the largest residual DCT coefficients in the frame under consideration. At the decoder, bit-planes are decoded and the residual DCT coefficient are reconstructed by placing each bit at its respective positions. The quality of the reconstructed image increases with the number of bit-planes decoded, however there will be only one enhancement layer reconstructed.

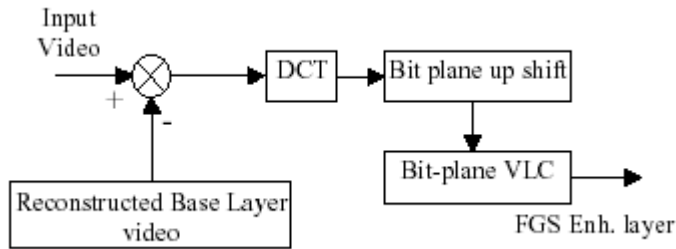


Figure 20: *FGS enhancement layer coding*

Variable rate FEC can be obtained using rate compatible punctured convolutional (RCPC) codes. As mentioned earlier, they provide a low complexity solution to variable rate FEC. As a result, allocation of bandwidth between source and channel coding can be effectively implemented using FGS compression and RCPC codes.

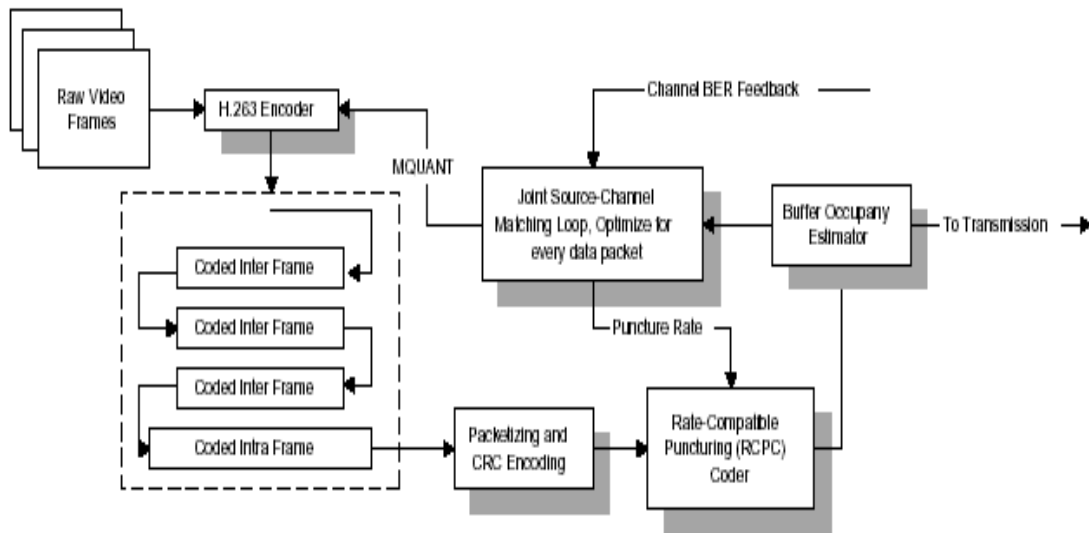


Figure 21: System for H.263 and RCPC Encoder

8 Visualization toolkit for simulating networks in ns2

8.1 Network Simulator ns2

Network simulator **ns2** is an event-driven network simulator used for networking research [18]. It is a widely used tool for simulating internetwork topologies to test and evaluate various networking protocols. There is a substantial support and flexibility in **ns2** to simulate various traffic generation patterns, routing and multicast protocols.

In order to study different networking issues like protocol interaction, congestion control, effect of network dynamics, scalability etc. it is necessary to simulate various scenarios that include different topology sizes, density distribution, traffic generation, membership distribution, real-time variance of membership, network dynamics etc. The **ns2** scenario generator can be used to create different random scenarios for simulation.

In **ns2**, characteristics of physical media of communication like delay, bandwidth, error rate, antennas and wireless physical interface parameters etc. can be defined. This helps in making the simulation studies as close to realistic scenarios as possible. **ns2** provides the flexibility to add and experiment new protocols or ideas. Recently, much support has been added for simulating wireless networks and interconnecting wired and wireless networks. Trace support in **ns2** may be used to trace packets for wireless and wired scenarios.

8.2 Emulator

It is also possible to use Network Simulator 2 as an emulator. Currently emulator support is available for FreeBSD operating system only. Emulator can be used to connect the tool to a live network. When the emulator is used one must use realtime scheduler

instead of schedulers mentioned in the preceding chapter. It connects the simulation time to real time. If the scheduler falls behind emulation fails. The emulator has two modes. In *protocol mode* the emulator interprets received traffic (e.g. ICMP Echo). In *opaque mode* received data is not interpreted. The emulator is connected to live network on the IP level.

8.3 Support software

8.3.1 Nam – VINT/LBL Network Animator

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet trace data. It supports topology layout, packet level animation, and various data inspection tools.

Nam began at LBL. It has evolved substantially over the past few years. Nam development effort is now an ongoing collaboration with the VINT project. The first step to use nam is to produce a trace file. The trace file should contain topology information, for example nodes, links and packet traces. During an NS2 simulation, user can produce topology configurations, layout information and packet traces using tracing events in NS2.

When the trace file is generated, it is ready to be animated by nam. Upon startup, nam will read the trace file, create topology, pop up a window, do layout if it is necessary and then pause at the time of the first packet in the trace file. Nam provides control over many aspects of animation through its user interface.

Nam does animation using the following building blocks: node, link, queue, packet, agent and monitor. Nam user manual can be found from

<http://www.isi.edu/nsnam/nam/>

This manual page is mentioned to be incomplete.

8.3.2 XGraph – an animating plotting program

One part of the ns-allinone package is *xgraph*, a plotting program which can be used to create graphic representations of simulation results.

XGraph with animation is a modification of the popular XGraph plotting program written by David Harrison at UC Berkeley. The original program supported line plots (and restricted surface plots) on any X11 display, and had several useful features, including being able to zoom on a region with the mouse.

Paul Walker added two features to the original code. First, it does crude animation of data sets. The animation only pages through data sets in the order in which they are loaded. It is quite crude, but useful if all your data sets are in one file in time order, and are output at uniform times. Also, the code will take derivatives of your data numerically and display those derivatives in a new XGraph window. These new features have been made a couple of years ago, and have no longer been supported.

There are a few annoying problems which haven't been cleaned up yet, the most aggravating of which is a problem with handling expose events, which sometimes forces the window to animate or refresh four or five times when it is exposed. Also, the derivative routines have not been rigorously tested. Finally, the labels are occasionally wrong when doing log plots.

Further information of XGraph can be found via
<http://jean-luc.ncsa.uiuc.edu/Codes/xgraph>.

8.4 Performance

We evaluate the influence of the transmission errors on TCP performance. The cases of using ARQ and FEC are considered as a data link layer protocol with improving the reliable packet transmission. We adopt Reed Solomon as the FEC code. We assume the ARQ overhead to be 5%. To choose the number of packet retransmissions by ARQ, we change it and compare the TCP performance. It is shown in Figure 22, where the horizontal axis E_b/E_0 is a noise level which affects communication errors. As shown in the figure, increasing the number of packet retransmissions at ARQ is very limited. Figure 23 shows the comparative results for ARQ and FEC. In the case of ARQ, the number of packet retransmissions is set to be one. As shown in the figure, FEC is more effective than ARQ to prevent TCP throughput degradation. It can also be observed that there exists the optimal FEC parameter to achieve the best performance dependent on the noise level.

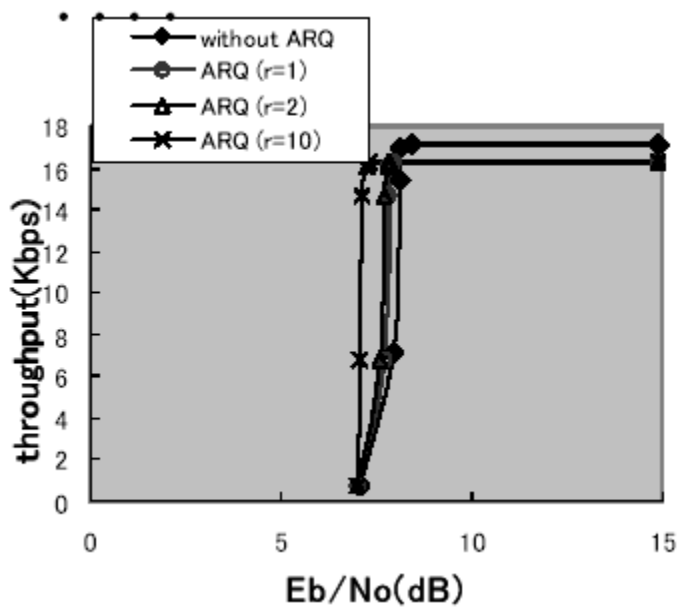


Figure 22: Influence of the number of packet retransmissions on TCP throughput

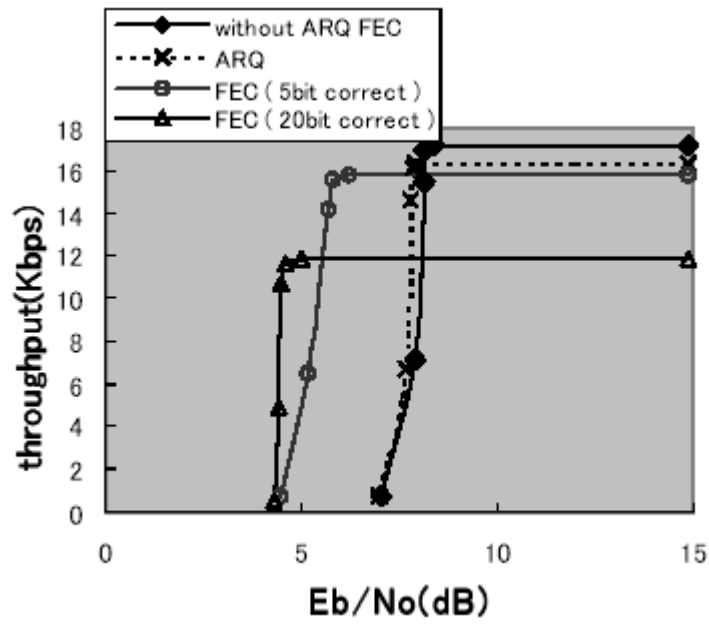


Figure 23: TCP throughput using ARQ, FEC

9. Program to demonstrate ARQ protocols

9.1 link.tcl

```
set link_delay 50ms
set link_bandwidth 1Mb
set simulation_time 100
set frame_size 1000
set ack_size 100
set send_timer 1
set ratio 0.1
set window 8
set ns [new Simulator]

$ns color 0 blue
$ns color 1 red
set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf

set n0 [$ns node]
set n1 [$ns node]
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"

$ns duplex-link $n0 $n1 $link_bandwidth $link_delay DropTail
$ns duplex-link-op $n0 $n1 orient right

source link_agents.tcl;

set link0 [new Agent/Message/StopWait]
$ns attach-agent $n0 $link0
set acker0 [new Agent/Message/StopWaitAcker]
$ns attach-agent $n1 $acker0
$acker0 set class_1
$ns connect $link0 $acker0

$link0 set packetSize_ $frame_size
$acker0 set packetSize_ $ack_size

set lossmodel0 [new ErrorModel/Uniform $ratio pkt]
set lossmodel1 [new ErrorModel/Uniform $ratio pkt]

$ns at 0.1 "$link0 start"
$ns at 0.1 "$acker0 start"

$ns at [expr 0.1 + $simulation_time] "finish"

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    puts "running nam..."
}
```

```

        exec nam out.nam &
        exit 0
    }
    $ns run

```

9.2 link_agents.tcl

```

source timer.tcl;
Class LinkTimer -superclass Timer

LinkTimer instproc set_callback callback {
    $self instvar callback_
    set callback_ $callback
}

LinkTimer instproc timeout {} {
    $self instvar callback_
    $callback_ timeout;
}

Class Agent/Message/StopWait -superclass Agent/Message

Agent/Message/StopWait instproc init args {
    $self instvar timer ;
    eval $self next $args ;
    set timer [new LinkTimer]
    $timer set_callback $self
}

Agent/Message/StopWait instproc start {} {
    global ns send_timer;
    $self instvar seq_no timer
    set seq_no 0
    $self sendframe;
}

Agent/Message/StopWait instproc sendframe {} {
    global ns send_timer
    $self instvar seq_no timer
    $self send $seq_no
    $ns trace-annotate "Send frame $seq_no"
}

Agent/Message/StopWait instproc handle msg {
    global ns send_timer
    $self instvar seq_no timer
    incr seq_no
    if {$seq_no > 1} {
        set seq_no 0
    }
    $self sendframe
}

Agent/Message/StopWait instproc timeout {} {
    $self sendframe
}

```

```
}
```

Class Agent/Message/StopWaitAcker -superclass Agent/Message

```
Agent/Message/StopWaitAcker instproc init args {  
    eval $self next $args  
    $self set_pkttype 5;  
}
```

```
Agent/Message/StopWaitAcker instproc start {} {  
    $self instvar seq_no  
    set seq_no 0  
}
```

```
Agent/Message/StopWaitAcker instproc handle msg {  
    global ns  
    $self instvar seq_no  
    $self send $msg  
    if { $seq_no == $msg } {  
        $ns trace-annotate "Ack frame $seq_no"  
        incr seq_no  
        if { $seq_no > 1 } {  
            set seq_no 0  
        }  
    } else {  
        $ns trace-annotate "Ignored received frame at [$ns now]"  
    }  
}
```

Class Agent/Message/GoBackN -superclass Agent/Message

```
Agent/Message/GoBackN instproc init args {  
    $self instvar timer ;  
    eval $self next $args ;  
    set timer [new LinkTimer]  
    $timer set_callback $self  
}
```

```
Agent/Message/GoBackN instproc start {} {  
    global ns send_timer window  
    $self instvar seq_no acked_seq_no send_window  
    set seq_no 0  
    set acked_seq_no -1  
    set send_window $window  
    $self trytosend; # Get started sending frames  
}
```

```
Agent/Message/GoBackN instproc trytosend {} {  
    $self instvar seq_no acked_seq_no send_window  
    while { $seq_no <= [expr $send_window + $acked_seq_no] } {  
        $self sendframe;  
        incr seq_no  
    }  
}
```

```

Agent/Message/GoBackN instproc sendframe {} {
    global ns send_timer
    $self instvar seq_no timer
    $self send $seq_no
    $ns trace-annotate "Send frame $seq_no"
    $timer sched $send_timer
}

```

```

Agent/Message/GoBackN instproc handle msg {
    global ns send_timer
    $self instvar acked_seq_no seq_no timer
}

```

```

Agent/Message/GoBackN instproc timeout {} {
    $self restart
}

```

```

Agent/Message/GoBackN instproc restart {} {
    $self instvar seq_no acked_seq_no
    set seq_no [expr $acked_seq_no + 1]
    $self trytosend
}

```

Class Agent/Message/GoBackNAcker -superclass Agent/Message

```

Agent/Message/GoBackNAcker instproc init args {
    eval $self next $args
    $self set_pkttype 5;
}

```

```

Agent/Message/GoBackNAcker instproc start {} {
    $self instvar seq_no
    set seq_no 0
}

```

```

Agent/Message/GoBackNAcker instproc handle msg {
    global ns
    $self instvar seq_no
    if { $msg < $seq_no } {
        $ns trace-annotate "Ack old frame $msg"
    } elseif { $msg == $seq_no } {
        $ns trace-annotate "Ack new frame $msg"
    } else {
        $ns trace-annotate "Discard new frame $msg"
    }
}

```

9.3 timer.tcl

Class Timer

```

Timer instproc sched delay {

```

```

        global ns
        $self instvar id_
        $self cancel
        set id_ [$ns at [expr [$ns now] + $delay] "$self timeout"]
    }

Timer instproc destroy {} {
    $self cancel
}

Timer instproc cancel {} {
    global ns
    $self instvar id_
    if [info exists id_] {
        $ns cancel $id_
        unset id_
    }
}

Timer instproc resched delay {
    $self sched $delay
}

Timer instproc expire {} {
    $self timeout
}

```

10 Conclusions

In this paper, we reviewed many tools that are used for efficient video transmission over wireless channels. Most of these tools can be used in combination with others. In many cases they offer significant improvements in performance with little overhead and complexity. Some of them are already included in current standards such as MPEG-4.

As potential applications are numerous and will be widely used, the research in the field is expected to continue. However, now the major obstacle is the quality vs. bandwidth trade-off, which is expected to be lessened by the introduction of new generation wireless systems.

The next challenge for video transmission may arise by the introduction of Internet protocol (IP) to wireless devices. In that case, IP related problems, such as network congestion, have to be faced together with current problems of wireless channels.

Appendix A

Reed-Solomon codes

The Reed-Solomon codes (RS codes) are non-binary cyclic codes with code symbols from a Galois field. They were discovered in 1960 by I. Reed and G. Solomon. The work was done when they were at MIT Laboratory. There are a number of ways to define Reed-Solomon codes. Reed and Solomon's initial definition focused on the evaluation of polynomials over the elements in a finite field. Others have preferred to examine RS codes in light of the Galois field Fourier transform. In the decades since their discovery, RS codes have enjoyed countless applications from compact discs and digital TV in living room to spacecraft and satellite in outer space.

The most important RS codes are codes with symbols from $GF(2^m)$. One of the most important features of RS codes is that the minimum distance of an (n, k) RS code is $n - k + 1$. Codes of this kind are called "maximum-distance-separable codes".

RS Codes with Symbols from $GF(2^m)$

- Let α be a primitive element in $GF(2^m)$.
- For any positive integer $1 \leq t \leq m$, there exists a t -symbol-error-correcting RS code with symbols from $GF(2^m)$ and the following parameters:

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

The generator polynomial is

$$g(x) = (x + \alpha)(x + \alpha^2) \dots (x + \alpha^{2^t}) = g_0 + g_1x + g_2x^2 + \dots + g_{2^t-1}x^{2^t-1} + x^{2^t}$$

Where $g_i \in GF(2^m)$ Note that $g(x)$ has $\alpha, \alpha^2 \dots \alpha^{2^t}$ as roots.

Example:

$$m = 8, t = 16, n = 255, k = n - 2t = 223, d_{\min} = 33$$

It is a (255, 223) RS code. This code is NASA standard code for satellite and space communications.

A.1 Reed-Solomon Encoder

Let $m(x) = m_0 + m_1x + \dots + m_{k-1}x^{k-1}$ be the message polynomial to be encoded. Where $m_i \in GF(2^m)$ and $k = n - 2t$.

Dividing $x^{2t}m(x)$ by $g(x)$, we have $x^{2t}m(x) = a(x)g(x) + b(x)$ where $b(x) = b_0 + b_1x + \dots + b_{2^t-1}x^{2^t-1}$ is the remainder. Then $b(x) + x^{2t}m(x)$ is the codeword polynomial for the message $m(x)$.

According to Dr. Meth[17], every code polynomial $c(x)$ can be expressed as $c(x) = m(x)g(x)$. If we give $m(x)$ as the message polynomial, the polynomial multiplication will encode it to a code word represented by $c(x)$. But in most cases, it is not systematic. To have the message bits occupy the last k positions of the code word will greatly simplify

the implementation of the decoder. A systematic encoding algorithm for an (n, k) cyclic code is

Step 1. Multiply the message polynomial $m(x)$ by x^{n-k} .

Step 2. Divide the result of Step 1 by the generator polynomial $g(x)$. Let $d(x)$ be the remainder.

Step 3. Set $c(x) = x^{n-k}m(x) - d(x)$.

Summarizing above, the following encoding procedure for RS codes is established.

The RS encoding procedure:

To construct a t -error correcting q -ary RS code of length n ($n = q - 1$):

1. Find a primitive n th root of unity α in $GF(q)$
2. Select $2t$ consecutive powers of α , starting with α^b (If $b = 1$, the code is narrow-sense RS).

Obtain the generator polynomial $g(x)$ as in (1.1)

3. Follow the 3 steps shown above for systematic encoding of a message given by $m(x)$

RS codes are spectrally efficient in the sense that they introduce less amount of redundancy, i.e., $r = n - k$. The redundancy introduced by RS is determined by the degree of $g(x)$, i.e., $r = 2t$.

RS codes are classified as maximum-distance separable (MDS) codes, as an (n, k) RS code always has minimum distance exactly equal to $(n - k + 1)$. MDS codes provide many design benefits, which is beyond the scope of this project and thus the discussion is omitted.

A.2 RS Codes for Binary Data

Every element in $GF(2^m)$ can be represented uniquely by a binary m -tuple, called a m -bit byte. Suppose an (n, k) RS code with symbols from $GF(2^m)$ is used for encoding binary data. A message of km bits is first divided into k m -bit bytes. Each m -bit byte is regarded as a symbol in $GF(2^m)$.

The k -byte message is then encoded into n -byte codeword based on the RS encoding rule. By doing this, we actually expand a RS code with symbols from $GF(2^m)$ into a binary (nm, km) linear code, called a binary RS code. Binary RS codes are very effective in correcting bursts of bit errors as long as no more than t bytes are affected.

A.3 Decoding of RS Codes

An important property of RS codes is that in practice they can be decoded rather easily, using GF arithmetic. Commercial RS encoder-decoders operating at speeds of up to 80Mb/s are available.

Thus, in decoding a RS code, we not only have to determine the error-locations but also have to evaluate the error values. If there are s erasure symbols and v errors in the received polynomial $r(x)$, then the (n, k) RS decoder corrects these erasure symbols and errors if $2v + s \leq d - 1 = n - k$. The received polynomial is represented by $r(x) = c(x) + e(x) + e^*(x) = c(x) + u(x)$, Where $e(x)$ and $e^*(x)$ are the error and erasure polynomials, respectively.

A.4 Errors-only Decoding of RS Codes

For errors-only decoding the received polynomial $r(x)$ has the simple form, r

$$r(x) = c(x) + e(x) \text{ where } e(x) = e_0 + e_1x + \dots + e_{n-1}x^{n-1}, e_i \in GF(2^m)$$

For $e_i \neq 0$, let e_i denote the error value at the i -th position. Suppose there are w errors, where $2w \leq n - k$ at positions i_w for $w = 1, 2, \dots, w$. The objective of the RS decoder is to find the number of errors, their positions and then their values.

References

1. S. Appadwedula, D. L. Jones, K. Ramchandran, and I. Konzintsev, "Joint sourcechannel matching for a wireless communications link," in *Proceedings of ICC 98*, Jun. 1998.
2. S. B. Z. Azami, O. Rioul, and P. Duhamel, "Performance bounds for joint source-channel coding of uniform memoryless sources using a binary decomposition," in *Proceedings of European Workshop on Emerging Techniques for Communication Terminals*, pp. 259–263, Jul. 1997.
3. B. Belzer, J. D. Villasenor, and B. Girod, "Joint source-channel coding of image with trellis coded quantization and convolutional codes," in *Proceedings of ICIP'95*, vol. 2, Aug. 1995.
4. M. Brystrom and J. W. Modestino, "Combined source channel coding for transmission of video over a slow-fading Rician channel," in *Proceedings of ICIP98*, 1998.
5. Po Rong Chang and Chin Feng Lin, "Wireless atm-based multicode CDMA transport architechure for MPEG-2 video transmission," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1807-1823, October 1999.
6. "Error Correction Codes," <http://imailab-www.iis.u-tokyo.ac.jp/Robert/codes.html>.
7. G. Davis and J. Danskin, "Joint source and channel coding for image transmission over lossy packet networks," in *SPIE Conference on Wavelet Applications of Digital Image Processing XIX*, Aug. 1996.
8. Robert E Van Dyck and David J Miller, "Transport of wireless video using separate, concatenatedm and join source-channel coding," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1734-1749, October 1999.
9. M. P. C. Fossorier, Z. Xiong, and K. Zeger, "Joint source-channel image coding for a power constrained noisy channel," in *Proceedings of ICIP98*, 1998.
10. Hamid Gharavi and Siavash M Alamouti, "Multipriority video transmission for third-generation wireless communication systems," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1751-1762, October 1999.
11. Berend Girod and Niko Farber, "Feedback-based error control for mobile video transmission," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1707-1722, October 1999.

12. Joachim Hagenauer and Thomas Stockhammer, "Channel coding and transmission aspects for wireless multimedia," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1764-1777, October 1999.
13. J. Hagenauer, "Rate-compatible punctured convolutional codes and their applications," *IEEE Transactions on Communications*, vol. 36, pp. 389-400, April 1988.
14. T.-H. Lan and A. H. Tew.k, "Power optimized mode selection for H.263 video coding and wireless communications," in *Proceedings of ICIP98*, 1998.
15. J. Lu, A. Nosratinia, and B. Aazhang, "Progressive source-channel coding of images over bursty error channels," in *Proceedings of ICIP98*, 1998.
16. H. Man, F. Kossentini, and M. Smith, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, pp. 8-11, Aug. 1997.
17. C. Meth, "Reed-Solomon Cells Now Licensable", *Electronic Design*, p. 171, Sept. 5, 1995.
18. "The network simulator -ns-2," <http://www.isi.edu/nsnam/ns/>.
19. "Packet Video," <http://www.packetvideo.com>.
20. K. Ramchandran, A. Ortega, K. Uz, and M. Vetterli, "Mutilresolution broadcast for digital HDTV using joint source/channel coding," *IEEE Journal on Selected Areas in Communications*, vol. 11, pp. 6-23, Jan. 1993.
21. G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, pp. 72-81, Jul. 1997.
22. John D Villasensor, Ya Qin Zhang, and Jiangtao Wen, "Robust video coding algorithms and systems," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1724-1732, OCTOBER 1999.
23. Yao Wang and Qin Fan Zhu, "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol.86, no. 5, pp. 974-997, May 1998.
24. H. Zheng and K. J. R. Liu, "Image and video transmission over wireless channel: A subband modulation approach," in *Proceedings of ICIP98*, 1998.