uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Babak Simaie

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Integrated Error Resilient Solutions for Motion JP2 Video Streaming

TITRE DE LA THÈSE / TITLE OF THESIS

A. Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

C-H. Lung

S. Shirmohammadi

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# Integrated Error Resilient Solutions for Motion JP2 Video Streaming

by

Babak Simaie

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

*Your file Votre référence*
*ISBN: 978-0-494-32478-3*
*Our file Notre référence*
*ISBN: 978-0-494-32478-3*

# Abstract

Two of the main challenges and critical issues facing real-time wireless multimedia transmission are packet loss and transmission errors. This thesis has selected Motion JPEG 2000, the newest standard for still image coding, for real time video streaming. JPEG2000 offers a number of qualities of service and revenue strategies that make it a leading contender for image streaming applications. We offer extra protection for the wireless transmission of JPEG2000 code streams due to error prone and changeable wireless channel conditions, and since the baseline JPEG 2000 error resilience tools are not sufficient in the context of wireless transmissions. Also, to accommodate the video streaming, avoid packet loss, and achieve efficiency, a payload format for JPEG 2000 video streams over the Real-time Transport Protocol has been defined. The main contribution of this thesis is adopting the packetization strategies and proposing an unequal error protection approach to improve the error resilience tools in baseline JPEG 2000 utilized in wireless streaming over the RTP. Our unequal error protection solution is directly supplied by a mechanism embedded in the JPEG 2000 syntax. At the end, the proposed approach is evaluated subjectively and objectively against existing wireless imaging techniques to demonstrate the achieved quality.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations:

| | |
|---|---|
| **ARQ** | Automatic Repeat-reQuest |
| **BCH** | Bose-Chaudhuri-Hochquenghem code |
| **BER** | Bit Error Ratio |
| **BPP** | Bits per Pixel |
| **CBR** | Constant Bit Rate |
| **CRC** | Cyclic Redundancy Check |
| **DWT** | Discrete Wavelet Transform |
| **DCT** | Discrete Cosine Transform |
| **DZQ** | Dead zone Scalar Quantization |
| **EBCOT** | Embedded Block Coding with Optimized Truncation |
| **FEC** | Forward Error Correction |
| **ICT** | Irreversible Color Transform |
| **IETF** | Internet Engineering Task Force |
| **JPEG** | Joint Photographic Experts Group |
| **JPWL** | JPEG 2000 Wireless |
| **MPEG** | Moving Picture Experts Group |
| **MSE** | Mean Square Error |
| **PSNR** | peak Signal-to-Noise Ratio |
| **RCPC** | Rate-Compatible Punctured Convolutional codes |
| **RCT** | Reversible Color Transform |
| **RS** | Reed Solomon codes |
| **RTP** | Real-time Transport Protocol |
| **RTCP** | Real-time Control Protocol |
| **SNR** | Signal-to-Noise Ratio |
| **SPIHT** | Set Partitioning in Hierarchical Trees |
| **ULP** | Unequal packet Loss Protection |
| **UEP** | Unequal Error Protection |

# Acknowledgements

I would like to thank Dr. Azzedine Boukerche for his supervision of my research work. Also I would like to thank Dr. Tingxue Huang and Richard Werner Nelem Pazzi for helping me with this thesis in choosing the best solution.

# Chapter 1

# Introduction

## 1. Historical Perspective

To date, some of the most innovative still image compression standards have resulted from the ongoing work of the ISO/IEC group of experts, the so-called Photographic Experts Group (JPEG). JPEG2000 is the most recent addition to a family of international standards developed by the Joint Photographic Experts Group, hereinafter referred to simply as WG1.

With the huge expansion of the Internet and its multimedia applications, the needs and requirements of the internet technologies have evolved and grown. Early in 1996, Dr. Daniel Lee of HP Company was named the WG1 Convener. He oversaw the development of a proposal [29] which resulted in the approval of JPEG2000 as a new WG1 work item. In March 1997, a call [12] for technical contributions was issued, requesting that compression technologies be submitted to an evaluation during the November 1997 WG1 meeting in Sydney, Australia. This call for contributions included a copy of the original proposal desiring unique feature set such as error resilience, scalability and ROI in JPEG2000, all within a single, tightly integrated compression architecture and code stream syntax. The standard was intended to complement and not to replace the current JPEG standards. Also, one of the aims of the standardization committee was the development of Part I of the JPEG2000 standard, which could be used on a royalty and fee free basis.

Finally, the standardization process, which is orchestrated by the WG1 of ISO/IEC, devised the Final Draft International Standard (FDIS) in August 2000 [8]. Furthermore International Standard (IS) was scheduled for December 2000 for editorial changes, and therefore there were no more technical or functional changes in Part I of the Standard.

Among the features, robustness to transmission bit-errors is one of the most desired features for JPEG2000 in the context of the wireless streaming, and makes it a leading contender for wireless multimedia applications. Given the importance of wireless

imaging applications, JPEG2000 kicked off a new activity referred to as Wireless JPEG 2000 or JPWL, known formally as part 11 of the JPEG 2000 standard [28]. It aims to transmit JPEG 2000 image data over an error-prone wireless transmission environment by extending the baseline specifications. More specifically, JPWL extends the specifications in the baseline of the JPEG2000 standard (Part I) with mechanisms for error protection and correction. These extensions are backward compatible in the sense that decoders that implement JPEG2000 baseline are able to skip and ignore the extension elements defined in JPWL.

The techniques introduced by JPWL yield a reasonable level of robustness, but they may still fail to guarantee acceptable quality in harsh transmission contexts such as the wireless one. The use of forward error correction codes in unequal error protection architecture, packetization, and interleaving in order to protect the bit stream is a common practice. Also some tools are designed to accommodate specific application requirements. They mostly established a trade-off between packet loss robustness and bit level error protection.

Our work is aimed at overcoming packet loss and residual bit errors in the context of wireless imaging and is implemented over the RTP standard. Since the introduction of the RTP standard (RFC 1889) in 1996 [45], it has been adopted by several standard organizations as the primary standard for real time multimedia transport over IP networks. The RTP protocol opens a new era of network multimedia where each node benefits from real time streaming multimedia and its applications. RTP was also published by the ITU-T as H.225.0, but was later removed once the IETF had a stable standards-track RFC published. It exists as an Internet Standard (STD 64) defined in RFC 3550 [46] (which made obsolete RFC 1889). In RFC3550, compared to REC1889, there are no changes in the packet formats on the wire, only changes to the rules and algorithms governing how the protocol is used. The biggest change is an enhancement to the scalable timer algorithm for calculating when to send RTCP packets in order to minimize transmission in excess of the intended rate when many participants join a session simultaneously.

There exist several efforts to define JPEG 2000 Video Streams (Motion JPEG2000 [24]) as RTP Payload [15]; however, they do not address error robustness in the presence of

errors in the bit stream and the packets consequently lost. In the subsequent chapters, our proposed algorithmic methods that enable error resilience of the JPEG2000 video stream using RTP over wireless channel are described.

# 2. Motivation and Scope of the Thesis

With the rapid developments in wireless communications and multimedia systems, video streaming and image compression techniques have become more important for wireless application. One of the challenges and critical issues faced by wireless application developers, regarding streaming wireless multimedia transmission is transmission error.

Given the goal of error detection and correction many efforts have been invested to propose algorithms to build error resilience into the compression techniques to protect stream data against error prone wireless channels and congested networks. Of course, the corrupted or lost data can always be re-transmitted to achieve error free data; however, a stream oriented network such as a mobile communication network imposes a low delay constraint that makes this approach impractical. Besides, the challenge of robust transmission is to protect digital data against hostile channel conditions and situations while removing excessive redundancies and achieving high bandwidth efficiency for limited capacity wireless environments.

Instead, it would be desirable to enable the receiver to correct errors on the fly in an incoming transmission on the basis of the bits in that transmission. To accommodate data stream protection, on the transmission end each k-bit block of data is mapped into an n-bit block ($n > k$) called a codeword, using an FEC encoder. The codeword is then transmitted; in our case (wireless transmission) a modulator produces an analog signal for transmission. During transmission the signal is subject to noise, which may produce bit errors in the signal. At the receiver, the incoming signal is demodulated to produce a bit string that is similar to the original codeword but may contain errors. This block is passed through an FEC decoder in coordination with the encoder FEC with one of four possible outcomes: i) there is no bit error and the decoder produces the original data block as output; ii) the decoder is able to detect and correct errors (certain patterns); iii) the decoder can detect but not correct the errors (certain patterns), and thus simply marks an uncorrectable error (residual errors); and iv) the decoder does not detect that any errors

have occurred (possible and rare patterns).

It is also important to see that wireless image streaming utilizes the Internet for real time image delivery and is subject to network congestion and packet loss. Thus, the core approach to protecting and correcting the data stream is on the basis of the packetization of the image data to control the end-to-end data flow.

This thesis gives an account of my investigations of the integration of error resilient solutions into the real time wireless streaming video. I utilize JPEG2000, a new state-of-the-art ISO/ITU still image compression standard to present a video stream that is formed by extending a single image into a series of JPEG 2000 images. We also offer RTP/ UDP-Lite/IP/MAC as underlying protocols to deliver real time video streams from the server to the client over a wireless network.

I define a payload format for JPEG 2000 video streams over the Real-time Transport Protocol and present the format to incorporate JPEG2000 error resilient schemes at both the source and packet levels into the RTP-based transmission of JPEG2000 wireless imaging. Four new marker segments are added to the final JPEG2000 code stream to indicate the error resilient schemes used in the code stream. The EPC marker segment indicates which JPEG2000 normative and informative tools are used in the code stream. More specifically, EPC signals whether the three other normative marker segments defined by JPWL. The primary function of the EPB marker is to protect the Main and Tile-part header. The ESD marker segment contains information about the sensitivity of the code stream to errors, and the RED marker segment signals the presence of residual errors in the code stream.

Configuration of UDP-lite/IP/MAC is beyond the scope of the thesis; however, the necessary explanation of the standards relevant to our proposal is presented throughout the thesis and clarified for completeness purposes.

By applying the mentioned tools and techniques, low delay robust transmission of video stream and high bandwidth efficiency is achieved and provides the basis for designing a system to effectively overcome the abrupt change of channel conditions, allowing developers to design and offer a wide range of Quality of Service strategies.

Finally in the experimental results section the superiority of our proposal over different standards is objectively (PSNR) and subjectively (Image) judged in various bpp or

compression ratios by Graphs and figures. We also compare the achieved fidelity of motion JPEG2000 with regards to MPEG4/H264 and conclude the benefits of our proposal. At the end the performance of different error resilient tools has been evaluated to improve the image quality in presence of random and burst error types in order to establish a suitable trade off between redundancy and the protection level.

# 3. Overview and Contribution of the Thesis

The widely used video MPEG4 [25] encoder is a standard primarily to compress audio and video (AV) digital data. Introduced in late 1998, it is the designation for a group of audio and video coding standards and related technology agreed upon by the ISO/IEC Moving Picture Experts Group (MPEG). MPEG4 is based on temporal redundancies or inter-frame coding and uses block-based DCT for data transformation. It employs motion estimation and compensation among frames to remove temporal redundancy and achieves high coding efficiency.

Drawbacks of this approach are high computational complexity resulting from motion estimation and compensation process and dependencies between coded frames introduced by the temporal prediction loop. Therefore, the negative impact of transmission errors propagates across several frames. Finally, due to the intra-/inter-frame coding structure and the inherently inaccurate rate control, coding delay is introduced.

Likewise, our proposal makes use of JPEG 2000 video stream or MotionJPEG2000 for wireless video streaming to offer more intra-frame efficiency and resilience and less coding delay and complexity.

As noted, in terms of error resiliency, Motion JPEG2000 has the upper hand over MPEG-4, as the impact of transmission errors is limited to a single frame and does not propagate to subsequent frames. Besides, each individual frame or image of compressed JPEG2000 is robust to bit errors introduced by noisy communication channel. This goal has been met by the inclusion of resynchronization markers as segmentation symbols "1010", the coding of data in relatively small independent blocks, the termination of the arithmetic coder for each coding pass or segmentation, and the provision of mechanisms to detect and conceal errors within each block. The standard also supports code-stream organizations, which can prove advantageous to applications requiring substantial error

resilience.

Yet another benefit of our approach is superior intra-frame coding efficiency. At high bit-rates, where artifacts become just imperceptible, JPEG2000 has a compression advantage over JPEG (intra-frame coding of MPEG-4) of roughly 20% on average in terms of ratio and fidelity. At lower bit-rates, JPEG2000 has a much more significant advantage over certain modes of JPEG. In addition, Motion JPEG2000 has lower coding efficiency at the benefit of lower computational complexity due to the use of inter-frame alone.

Our proposed JPEG 2000 video stream makes use of a precise post-compression rate control technique (the inclusion of a selected subset of coding passes in the code stream). Alternatively, rate control can be achieved by the choice of quantizer step sizes, but this technique has a potential drawback in spite of its simplicity. Whenever the quantizer step sizes are adjusted, the quantizer indices change, and tier-1 encoding must be employed again. Since tier-1 coding requires a considerable amount of computation, this approach to rate control may not be practical in computationally constrained encoders such as wireless handheld devices. As a direct consequence, the first method has been chosen to achieve a precise target bit rate, thus introducing minimal coding delays.

Conversely, MPEG-4 uses the classical paradigm of rate control based on modeling and feedback, which is inherently inaccurate. Furthermore, the intra-/inter frame coding structure (I-, P- and B-frames) gives different importance to frames. As a result, bit rate often fluctuates greatly from one frame to another, resulting in increased buffering and coding delay.

Also in our approach, the multi-resolution wavelet coding and embedded bit stream in Motion JPEG 2000 allows for a very efficient resolution or quality scalability leads to transmit data progressively. Progressive transmission is highly desirable when receiving imagery over slow communication links. Code-stream organizations that are progressive by "component" still improve the quality of decoded imagery as more data are received. In case of packet error or loss, code stream organizations which are progressive by "component" facilitate the resynchronization of the packets. The efficiency of the code stream improves dramatically with optimal discarding of the damaged data and ultimately the quality of the output video stream improves as well.

In a compressed JPEG2000 code stream, some parts of the data (header and lower layers)

are more important than others. In these cases it is sometimes possible to get most of the effect while recovering only part of the packet. JPEG2000 has a minimum number of bits that need to be recovered to avoid decoder crash and provide a visible image, with additional bits that are not essential but improve the image quality if they can be recovered. A recovery scheme that recovers only the minimum data will be lower in quality than one that recovers the complete packet, but it may have significantly less overhead.

Our solution gives the most important part of the packet a greater degree of protection. In this case the entire packet is recovered with some probability, but the important parts have a higher chance of recovery. We exploit Reed–Solomon codes, which are an alternative to parity codes that offer protection with less bandwidth overhead at the expense of additional complexity. In particular, they offer good protection against burst loss, where conventional parity codes are less efficient.

For the purpose of streaming data over Internet, the final JPEG2000 code stream is packetized and wisely inserted into the RTP packets and fragmented as needed. The RTP headers are defined and assigned appropriate value for consistency, and error and packet loss protection.

Overall, the contribution this thesis can make is the integration of error resilient solutions (bit error and packet loss correction and protection) into the RTP-based transmission of JPEG2000 wireless imaging, by taking advantage of the progressive scalability, high coding efficiency, low delay and computational complexity, and open architecture of JPEG2000.

# 4. Organization of the Thesis

The remainder of the thesis is organized as follows: Chapter 2 describes a generic digital communication system and provides background information on robust data stream communication error resilient coding algorithms consisting of Source/channel and Joint Source Channel Coding, and also a robust framework for transporting real-time video.

In Chapter 3, we briefly explain JPEG 2000, A New Paradigm in Image Compression, its final Code stream format, baseline error resilient tools, and RTP, which is utilized to transport JPEG2000 video stream. Moreover, we specify a payload format for JPEG 2000

video streams over the RTP.

Chapter 4 deals with the error protection of RTP-packets containing JPEG2000 video streams. We investigate the robust real time transmission of JPEG 2000 packets including Unequal Packet Loss/error protection and data partitioning and interleaving.

In Chapter 5 we present experimental results that show the JPEG2000 performance study and the superiority of our approach in improving the PSNR of the final data stream.

Finally, Chapter 6 draws conclusions and explores approaches and directions for future work.

# Chapter 2

# Background

## 2.1. Introduction

In this chapter we provide the background of our work. To begin with, in section 2.2, we present the generic communication system structure and explain the components of the system. Section 2.3 overviews the various types of error resilient coding and their categories to incorporate quality of service into the delivery of real time multimedia over error prone communication channels and achieve the best possible throughput. Considering the desirable application, in section 2.4 we give a brief description of a framework for the robust transportation of real-time video over wireless networks.

## 2.2. Digital Communication System

Digital communication systems transmit the digital information from source(s) to one or more destinations through communication channels. Figure 2.1 demonstrates functional block diagrams of a generic communication system.

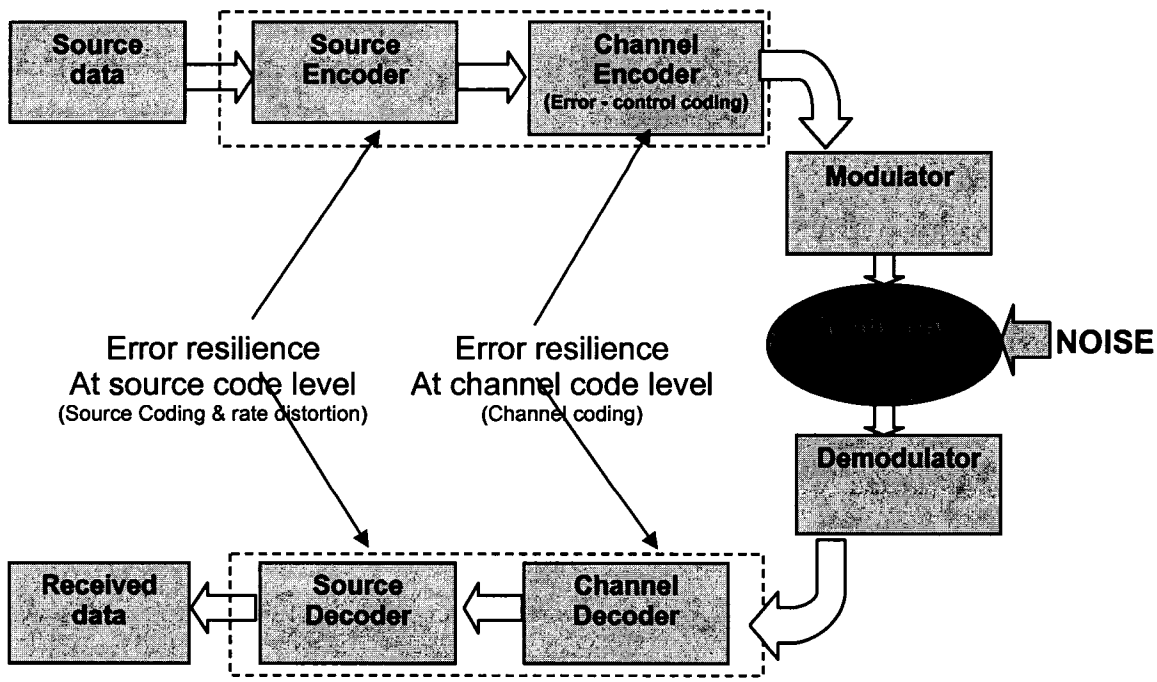



Figure 2.1: A General Framework for a Digital Communication System

As shown in Figure 2.1, the typical system consists of three modules working in serial at the sender side and three inverse modules at the receiver side. The ultimate goal of the system is to deliver error free source data at the best possible quality and throughput with the least possible delay and complexity. To advance the aim of the system two level of coding - Source and Channel - work in serial to provide different application to respond to different client needs. Following, we describe the system components and functionality:

- **Source Encoder:** This part of the system simply converts the source data into a sequence of bits or bit stream where the encoder performs data compression by removing redundancy as efficiently as possible. In a source digital data form, perhaps as a result of an analog-to-digital conversion step, redundancy presents in the data results from the fact that the number of bits used to store the information may exceed the amount of actual information content.

  The amount that a particular source of data can be compressed without any loss of information (*lossless* compression) is governed theoretically by the source coding theorem of information theory [49]; It states that a source of information can be represented without any loss of information in a such a way that the amount of storage required (in bits) is equal to the amount of information content. To achieve this lower bound, it may be necessary for long blocks of data to be jointly encoded. The source encoder employs special types of codes to do the data compression, called data compression techniques. There are four methods for compression: discrete cosine transform (DCT), vector quantization (VQ), fractal compression [31], and discrete wavelet transform (DWT), followed by Huffman coding, run-length coding, arithmetic coding, Lempel-Ziv coding and combinations of these, all of which fall beyond the scope of this thesis.

  If the data needs to be compressed below the information content rate of the source then some kind of distortion must occur. This is called lossy data compression.

  It is possible to do lossy compression in a way that minimizes the amount of distortion for a given rate of transmission. The theoretical limits of lossy data compression are established by the rate-distortion theorem of information theory; it addresses the problem of determining the minimal amount of entropy (or

information) R that should be communicated over a channel, so that the source (input signal) can be approximately reconstructed at the receiver (output signal) without exceeding a given distortion D. The rate is usually given in bits per pixel (bpp) for images, and in bits per second (bps) for video. An alternative to the bit rate is the compression ratio, which is the size of the compressed image in bits normalized by the number of bits that represent the original image. A common objective measure of the distortion is the mean square error (MSE), which is the mean of the squared errors between every pixel in the original image and the correspondent pixel in the reconstructed image. Another important measure of the reconstructed image quality is the peak signal-to-noise ratio (in dB), which can be derived for grey-scale images with a color depth of 8 bpp from the MSE as:

$$PSNR = 10\log_{10}\frac{255^2}{MSE}$$

The typical shape of R (D) looks like this where the R (D) function for a memory-less Gaussian source, with a variance $\sigma^2$ is:

$$R(D) = \begin{cases} \frac{1}{2}\log_2^{\sigma^2/D}, D \le \sigma^2 \\ 0, D > \sigma^2 \end{cases}$$

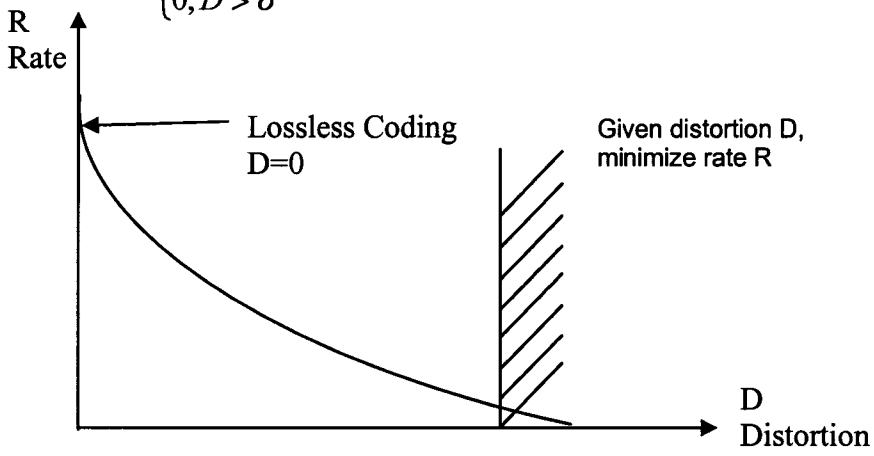



Figure 2.2: Rate/Distortion Graph

Source coding is the first step in the error correction and detection process; however, many coding techniques are not error resilient by nature and suffer from a lack of error detection and protection mechanisms. Likewise, JPEG2000, the new ISO/ITU-T standard for still image coding, presents built-in mechanisms that

aim to detect whenever an error occurs, conceal the erroneous data, and resynchronize the decoder.

- **Channel Encoder:** This encoder serves as an error control-coding module that adds additional bits to information bits to detect or even correct transmission errors. The error control-coding task can be either employed in the source encoder as indicated before as an integrated part of it, and/or in the channel encoder. The latter choice is called channel coding, which is a viable method to reduce the information rate through the channel and increase reliability. This goal is achieved by adding redundancy to the information symbol vector resulting in a longer coded vector of symbols that are distinguishable at the output of the channel.

  The former error control-coding method at the source level is integrated with the core encoding algorithm and proved to be more efficient and impose less redundancy and complexity. However, given the fact that various applications use different communication channels, this method must handle a wide variety of errors and, more importantly, the common encoding (source coding) algorithms are not error resilient by nature.

  As we will discuss later, the combination of these two approaches reduces the error resilience redundancy while maintaining the same error resilience efficiency [37] [17] [43].

  In fact, the error resilience efficiency is maintained by applying error resiliency at the source level to take care of residual errors where channel coding imposes excessive overhead to fix it. Also by using these tools at the source coding level we leave it to implementers to choose the best tool possible for the desirable application, thus improving product efficiency and differentiation.
- **Modulator/Demodulator:** These parts of the communication modulate the signal to overcome signal transmission issues, which are addressed in physical and MAC layer.
- **Channel Decoder:** This part exploits the redundancy of received information bits inserted by the channel encoder to detect and possibly correct transmission errors, discard and conceal the erroneous data, or mark the residual errors.
- **Source Decoder:** This section decompresses the data based on the original compression scheme at the source sender.

The generic digital communication system depicts the necessity of the robust transmission of the data from source to the destination. Figure 2.1 illustrates the implementation of error resilient tools at the source code and/or channel code. The proper design and implementation of error resilient tools result in minimal code overhead and increased reliability.

## 2.3. Error Resilience

One of the main requirements for wireless video transmission is error resilience. When transmitted over a wireless channel, video can be affected by a number of loss mechanisms, like multipath fading, shadowing, and co-channel interface. The effects of such errors are magnified due to the fact that the video bit stream is highly compressed to meet stringent bandwidth limitations. The higher the compression, the more sensitive the bit stream is to the errors, since in this case each bit represents a larger amount of decoded video. It is therefore not difficult to realize that when transmitted over a wireless channel, compressed video can suffer severe degradation, making the use of error-resilience techniques vital.

Generally the frame work of robust and error resilience data communication for real time video consists of two categories. Congestion control is about controlling traffic entry into telecommunication networks to avoid link overflowing by sending packets more slowly, while error control can be achieved through: 1-error resilience at source code level 2-channel coding 3-joint source/channel level.

The focus of this thesis is error resilience; however, for the sake of competency we briefly mention the congestion control methods and its effect on video stream robustness. In this section we briefly describe each error resilience category and cite the tools for each one as well as elaborate the techniques used in that category. Finally, we compare the tools of each class and explain the pros and cons of each method over the others.

### 2.3.1. Source Coding Error Resilience

Error resilient tools at the source coding level, called error resilient coding, is arguably proved to be more efficient than channel coding; however, joint source-channel coding usually achieves better error resilience than that of separate source or channel coding.

Generally, error resilient coding faces a broad range of error situation such as packet loss, of random or burst type, and needs to support different applications and a wide variety of communication channels. Some of the coding schemes that address these challenges can be found in literature. The most desirable scheme is the one that easily incorporates the error resilient scheme into the coding algorithms without an impact on coding efficiency, extensive redundancy, and complexity. The scheme should be scalable to adapt to varying channel conditions quickly and effectively to allow for a trade-off between redundancy and robustness.

One example of these techniques is the use of "elementary embedded bit stream", or briefly "embedded bit stream", where the bit stream data is ordered according to its importance and the need for error protection is decreased from the beginning to the end of the stream.

By nature, many coding schemes are not error resilient. JPEG2000 [26], a new international standard for still image compression has numerous features including source coding error resilience. This goal has been met by utilizing the natural prioritization of information induced by embedded block coding and quality layers, the inclusion of resynchronization markers (SOP), the coding of data in relatively small independent blocks (block code) and defining smaller precincts to result in smaller packets and the provision of mechanisms to detect and conceal errors within code blocks. One example of these methods can be the byte-stuffing procedure where the JPEG2000 arithmetic coder does not produce certain values (0xFF90 through 0xFFFF) inside the coding passes. The unexpected detection of one of these values would indicate that an error has occurred.

Finally optional code stream organizations by developers such as different progression are supported by the standard, which can prove advantageous to wireless applications requiring substantial error resilience.

### 2.3.2. Channel Coding

Error correction and detection has great practical importance in the digital communication field. Error detection is the ability to detect errors that are made due to noise or other impairments in the course of the transmission from the transmitter to the receiver. Error correction has the additional feature that it enables the localization and correction of the errors.

The two most practical approaches to Error Correction/Detection are defined as FEC and Error Detection/ARQ. The latter approach is based on receiver feedback or requests to re-send a block of data when errors are detected. Typically, such codes are used in conjunction with a protocol at the data link, Internet, or transport level. With an ARQ scheme, a receiver discards a block of data in which an error is detected and the receiver requests the transmitter to retransmit that block of data. The process of the retransmission of data proceeds until the receipt of correct data or a delay deadline is over.

Likewise, the FEC or channel coding, which are often used interchangeably, does not requires any feedback line and relies on additional data added by the sender to a media stream, which the receivers can then use to detect and possibly correct errors with a certain probability as the data is received.

### 2.3.2.1.. Error Detection/ARQ

One of the simplest error detection/correction codes is the parity code. The parity operation can be described mathematically as an exclusive-or (***XOR***) of the bit stream:

$$\forall A, B$$
$$A \oplus (A \oplus B) = B$$

This means that if we somehow transmit the three pieces of information the ***A***, ***B*** data and ***(A XOR B)*** parity separately, we need only receive two of the three pieces to recover the values of A and B. In the case that any even numbers of bits are inverted due to error, an undetected error occurs. Hence the parity bit is not foolproof as the noise impulse is often long enough to destroy more than one bit, particularly at higher data rates.

The better and most powerful detection approach is CRC. It is a type of hash function used to produce a checksum – a small, fixed number of bits – against a block of data, such as a packet of network traffic. The transmitter appends the checksum to the $k$ bits data such that the resulting frame, consisting of $n$ bits, is exactly divisible by some predetermined number. If the result has no remainder, the receiver assumes that no error has occurred. To formulate the approach, we define:

**T**= $n$-bit frame to be transmitted

**D** = $k$-bit block of data or message, the first $k$ bits of T

**F**= ($n$- $k$)-bit checksum, the last ($n$ - $k$) bits of T

**P**=pattern of $n$-$k$+1 bits; this is the predetermined divisor

It is clear that where checksum (F) is equal to the remainder (R):

$$T = 2^{n-k} D + R$$

Now we divide T by P to see if R satisfied the condition:

$$\frac{T}{P} = \frac{2^{n-k} D + R}{P} = Q + \frac{R}{P} + \frac{R}{P} = Q$$

Error detecting codes are largely used in communication systems. Some of the best known CRC codes are CRC-16-CCITT (X.25, V.41, Bluetooth, PPP, IrDA) and CRC-32 (Ethernet, FDDI, ZIP ,IEEE 802.3).

### 2.3.2.2. FEC

In the framework of wireless video streams if the source coding resilience tool is not enough when a channel has severe BER or fading and real-time constraint renders the ARQ approach impractical, FEC is necessary to have better visual quality.The FEC scheme generates a larger bit stream intended for transmission across a lossy medium or network. The additional information in the transformed bit stream allows receivers to exactly reconstruct the original bit stream in the presence of transmission errors. The two main classes of FEC codes are block codes and Convolutional codes.

Block coding is a way of encoding data in a communications channel that adds patterns of redundancy into the transmission path in order for the receiver to correct errors in an incoming transmission on the basis of the bits in that transmission and to lower the error rate. Redundancy in information theory is the number of bits used to transmit a message minus the number of bits of actual information in the message. Because of the redundancy introduced by the channel coder, there must be more symbols at the output of the coder than at the input. A channel coder operates by accepting a block of $k$ input symbols and producing at its output a block of $n$ symbols, with $n>k$. Commonly, these codes are referred to as $(n, k)$ block codes. The main characterization of a block code is that it is a fixed length channel code. Some of the commonly used block codes are Hamming codes, Golay codes, BCH codes, and Reed Solomon codes (which uses non binary symbols).

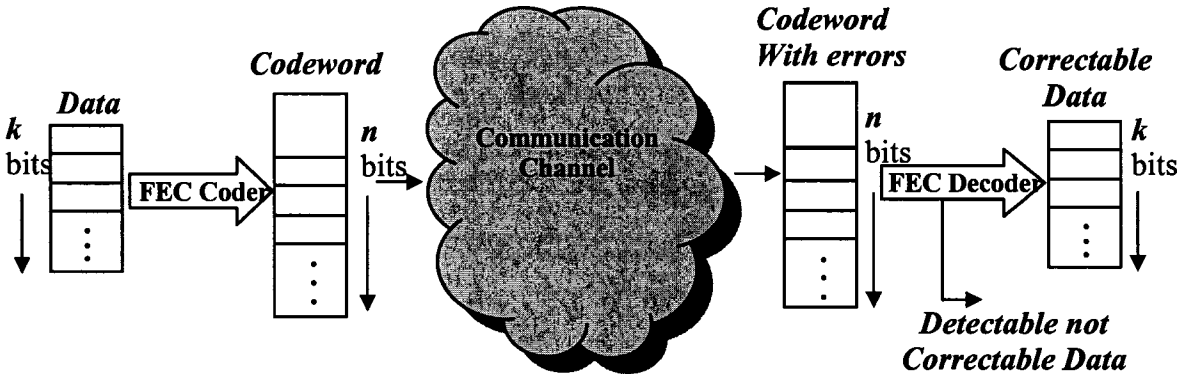



Figure 2.3: Forward Error Correction/Detection Process

We note that in many cases, to correct the errors in received data, the block code first detects the errors exploiting detection codes such as CRC in cyclic code and then performing additional processing for error correction.

The correction process makes use of the Hamming distance between two sequences of bits of equal length. The Hamming distance $d(V_1,V_2)$ between two ***n***-bit binary sequences $V_1$ and $V_2$ is the number of bits in which $V_1$ and $V_2$ disagree. For example, if $V_1 = 011011$, $V_2 = 110001$ then $d(V_1,V_2) = 3$.

Now considering the block code technique for error correction, we map each ***k***-bit sequence into a unique ***n***-bit codeword. Typically, each valid codeword reproduces the original ***k*** data bits and adds to them ***(n - k)*** check bits to form the ***n***-bit codeword. Thus the design of a block code is equivalent to the design of a function of the form $V_c = f(V_d)$, where $V_d$ is a vector of ***k*** data bits and $V_c$ is a vector of ***n*** codeword bits.

With an ***(n, k)*** block code, there are $2^k$ valid codewords out of a total of $2^n$ possible codewords. The ratio of redundant bits to data bits, ***(n - k)/ k***, is called the redundancy of the code, and the ratio of data bits to total bits, ***k/n***, is called the code rate. The code rate is a measure of how much additional bandwidth is required to carry data at the same data rate as it would without the code. For example, a code rate of ½ requires double the transmission capacity of an uncoded system to maintain the same data rate. For a code consisting of the codewords w1, w2... ws, where $s = 2^k$, the minimum distance $d_{min}$ of the code is defined as:

$$d_{min} = \min_{i \neq j} [(wi, wj)]$$

It can be shown that the following conditions hold. For a given positive integer t, if a code satisfies $d_{\min} \geq 2t+1$, then the code can correct all bit errors up to and including errors of t bits. If $d_{\min} \geq 2t$ then all $errors \leq t-1$ bits can be corrected, and errors of t bits can be detected but not, in general, corrected. Conversely, any code for which all errors of magnitude $\leq t$ are corrected must satisfy $d_{\min} \geq 2t+1$, and any code for which all errors of magnitude magnitude $\leq t-1$ are corrected and all errors of magnitude t are detected must satisfy $d_{\min} \geq 2t$.

Another way of putting the relationship between $d_{\min}$ and t is to say that the maximum number of guaranteed correctable errors per codeword satisfies:

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

Furthermore, if we are concerned only with error detection and not error correction, then the number of errors, t, that can be detected satisfies:

$$t = d_{\min} - 1$$

To see this, consider that if $d_{\min}$ errors occur, this could change one valid codeword into another. Any number of errors less than $d_{\min}$ can not result in another valid codeword.

As a result, the design of a block code involves a number of considerations for coding efficiency:

1. For given values of ***n*** and ***k***, we would like the largest possible value of $d_{\min}$.
2. The code should be relatively easy to encode and decode, requiring minimal memory and processing time.

A Convolutional code is a type of error-correcting code commonly specified by three parameters; ***( n, k, m)*** and first each k bit of data to be encoded is transformed into an ***n*** bit output, where ***k/n*** called the code rate ***(n >=k)*** and secondly the transformation is a function of the ***m*** which designates the number of previous ***k***-bit input blocks that must be memorized in the encoder.

***n*** = number of output bits

***k*** = number of input bits

***m*** =number of memory registers

The code rate is a measure of the efficiency of the code. Commonly, $k$ and $n$ parameters range from 1 to 8, $m$ from 2 to 10, and the code rate from 1/8 to 7/8 except for deep space applications where code rates as low as 1/100 or even longer have been employed.

Convolutional codes tend to operate on smaller blocks of data than block codes and, unlike block codes, the encoding of one block of data depends on the state of the encoder as well as on the data to be encoded. The decoding of convolutional codes is usually done by executing some type of decoding algorithm in a processor. With block codes, each block of data can be long and is encoded and decoded independently from every other block of data.

Several algorithms exist for decoding Convolutional codes. For relatively small values of $k$, the Viterbi algorithm is universally used. Simple Viterbi-decoded convolutional codes are now giving way to turbo codes [7], a new class of iterated short convolutional codes that closely approach the theoretical limits imposed by Shannon's theorem with much less decoding complexity than the Viterbi algorithm on the long convolutional codes that would be required for the same performance. Turbo codes have not yet been concatenated with solid (low complexity) Reed-Solomon error correction codes. However, in the interests of planetary exploration this may someday be done.

When a sender uses FEC, it must decide on the amount of FEC to add, on the basis of the loss characteristics of the network. One way of doing this is to look at the receiver feedback report packets it is getting back, and use the loss fraction statistics to decide on the amount of redundant data to include with the media stream.

### 2.3.3. Joint Source/Channel Level

The separation principle of Shannon states that the source coder and the channel coder can be separately designed without any loss of optimality. Furthermore, he showed that every communication channel is characterized by a quantity known as the channel capacity and that for any rate below channel capacity, a channel coder exists that reduces the error probability to any desired level. However, there are many practical problems in the way to construct coding schemes promised by Shannon. First, when operating under a delay-constraint on a time-varying channel, it is generally no longer optimal to regard the two coders separately and we have to jointly optimize the source coder and the channel

coder. Second, the separation principle implies that extremely large codes may have to be used, which complicates the implementation of such codes [54]

Thus many researchers have considered the combined design of the source coder and the channel coder, which is known as joint source/channel coding. This is a rather loose label that encompasses all coding techniques where the source and channel coders are not entirely separated.

Example of some different joint source/channel coding techniques include tandem structures with resource control, multi-resolution modulation, hybrid digital-analogue coders, and direct source-channel mappings both in the form of numerically optimized systems and parametric space-filling curves [22]. All indicated schemes have their strengths and weaknesses, and the choice of approach will depend on the given constraints.

## 2.4. Robust Framework For Transporting Real-Time Video

We focus our attention onto two prominent components in designing a robust streaming video system, namely error and congestion control. The congestion control relies mostly relying on network infrastructure and topology; on the other hand, the error coding control schemes employ techniques at the compression and channel coding level by adding on codes to networking protocols. In this section we are mainly concerned with controlling schemes with the objective of maximizing the likelihood of continuous video playback subject to varying channel conditions over wireless networks. Also, for the completeness of our discussion, we briefly explain the topic of error congestion control.

### 2.4.1. Error Control Coding

Error control coding is a huge research and engineering area where cutting edge pure mathematics has direct practical applications that are growing day by day. As indicated in section 2.3.2., the techniques used to control and correct transmission errors fall into two basic categories: forward error correction and retransmission. The choice between retransmission and forward error correction depends on the application and on network characteristics.

Automatic repeat-request (ARQ) is an error control technique used in data communication systems with feedback channels. Various techniques have been proposed that combine ARQ and error correcting codes to improve the performance of error control systems.

The hybrid ARQ schemes described above are known in literature under the names of Type I and Type II hybrid ARQ schemes. In Type I, upon detecting one or more errors in a transmitted codeword, the receiver first attempts to locate and correct them. If the errors can not be corrected by the given code, the receiver requests a retransmission of the codeword. This process is repeated until the codeword is successfully decoded.

In the Type II hybrid ARQ scheme [36], a rate 1/2 invertible error correcting code and an error detecting code is formed. The transmitter first computes the redundant sequence using the half-rate invertible code and the two sequences are then encoded by the error-detecting code. The scheme alternates the transmission of the encoded information symbols and the encoded redundant symbols, which are either detected as correct or combined for error correction using the half-rate invertible code. A more efficient Type II hybrid ARQ scheme [21] uses RCPC codes. The scheme is based on the successive transmission of redundant symbols until the RCPC code is strong enough to decode the transmitted data, instead of repeating the transmission of information or redundant symbols as in the previous scheme.

In spite of the good results yield by ARQ, for real time wireless applications this approach is inadequate mainly for three reasons:

1. The bit error rate on a wireless link can be quite high. This would result in a large number of retransmissions.
2. In some cases, especially in satellite links, the propagation delay is very long compared to the transmission time of a single frame. The result is a very insufficient system, whereas with a long data link, an error in a single frame necessitates retransmitting many frames.
3. Real-time applications often impose a low delay constraint, which renders the ARQ solution powerless due to its long data link

Consequently, real-time wireless applications have adopted the fast FEC approach to detect and correct errors on fly as the data comes in, and have thus established a trade off between the redundancy and complexity of the code.

## 2.4.2. Congestion Control

In an IP packet switch network, the stream multimedia shares the links with other traffic resulting in congestion when the total sending rate exceeds the link capacity. Potentially the situation may lead to congestion or even the collapse of the network where a sudden drop in delivery rate occurs. To prevent congestion the network layer makes its best effort to deliver the packets and discard excess packets if the link becomes congested. This process is called rate control, which attempts to minimize network congestion and the amount of packet loss by matching the rate of the video stream to the available network bandwidth

Rate-adaptive encoding, rate shaping, and rate control are different approaches designed to avoid congestion in a network. Note that rate control is from the transport perspective, while rate-adaptive video encoding is from the compression perspective,and rate shaping is in both the transport and compression domains. In this section, we give a brief description of the above mentioned rate control algorithms used in streaming video.

### 2.4.2.1. Channel Rate Control

For the constant bit rate channel, since the channel rate is fixed, the video rate must be constantly controlled and limited within the fixed channel capacity. For the variable bit rate channels, on the other hand, the quality of video is guaranteed by allocating the necessary bandwidth to achieve a given desired quality. However, network congestion may occur if the total bandwidth demand from multiple compressed video sources exceeds the channel capacity. In this case, video rate control strategy must be considered in order to maintain the video quality in an operational connection.

To control the channel rate, in a constant-rate channel, an encoder buffer is necessary to translate the variable rate output by the compression engine into the constant-rate channel. A similar buffer is also necessary at the receiver to convert the constant channel bit rate into a variable bit rate. Alternatively, compressed video can also be transmitted over variable-rate channels, e.g. broadband IP networks. These networks are able to support variable bit rates by partitioning video data into a sequence of packets and inputting them into the network asynchronously (channel rate control).

More specifically, congestion control on the Internet is implemented by the transport protocols layered above IP. It is a simple matter to explain the congestion control

functions of UDP—there are none unless they are implemented by the applications layered above UDP—but TCP has an extensive set of algorithms for congestion control [3][30].

#### 2.4.2.2. Rate-Adaptive Encoding

As noted before, the rate control can be achieved either through channel rate control or encoder rate control. In addition, the control techniques for the data rate of compressed video determine the encoding efficiency and video quality.

It is worth noticing that uncompressed video has a constant rate by nature and is transmitted over constant-rate channels, e.g. analog TV signal over terrestrial and cable broadcasting networks. Likewise, compressed digital video has a variable rate output since most video compression algorithms use variable length codes, e.g. Huffman codes. If a channel connection following the Variable length coder (Huffman) is CBR, buffering must be used to adapt the variable data rate to the fixed channel rate. Here it is desirable that rate control algorithms prevent buffer starvation and overflow at the decoder, both of which are critical to maintaining continuous video playback.

Different approaches are proposed for different compression schemes. For instance, in the emerging compression standard for image/video coding, the JPEG2000 encoder, rate control can be achieved through two distinct mechanisms:

1. the choice of quantizer step size
2. The selection of the subset of coding passes to include in the code stream.

These approaches are briefly described in the following chapter.

# Chapter 3

# Scalable and robust JPEG 2000 coding system for wireless streaming

## 3.1 Introduction

In the old days, most of the data carried over networks was textual data. Today, with the rise of multimedia and wireless network technologies, multimedia has become an indispensable and pervasive feature on the Internet. Streaming Voice and video clips have also become more and more popular on the Internet and wireless networks.

Wired multimedia networking is not a trivial task. We can expect at least three difficulties. First, compared with traditional textual applications, multimedia applications usually require much higher bandwidth. A typical 25 second 320 x 240 movie could take 2.3MB, which is equivalent to about 1000 screens of textual data. This would have been unimaginable in the old days when only textual data was transmitted over the net.

Second, streaming multimedia applications require real-time traffic. Audio and video data must be played back at receiver continuously at the rate at which they were sampled. If the data does not arrive in time, the playback will stop and human ears and eyes can easily pick up the artifact.

Third, a multimedia data stream is usually bursty. Just increasing the bandwidth will not solve the burstiness problem. For most multimedia applications, the receiver has a limited buffer. If no measure is taken to smooth the data stream, it may overflow or underflow the application buffer. When data arrives too quickly, the buffer will overflow and some data packets will be lost, resulting in poor quality. When data arrives too slowly, the buffer will underflow and the application will starve.

While significant progress is being made in developing a multimedia transmission infra-structure, there is by no means a guarantee that real-time wireless services, such as video transmission, will be easily deployed. For wireless multimedia, in addition to the above difficulties, the applications face new challenges in four areas. First, the low bandwidth of the wireless transmission link calls for a low or very low rate accomodated by efficient

compression algorithms, in particular through algorithms like Motion JPEG2000. In addition, it is obvious that the devices to be used for video display and capture have to be particularly efficient because they have to be portable. This places constraints on the memory and resources of these devices, thereby calling for compression algorithms that are optimized for compexity, memory, etc.

Given the variable nature of the channels we consider, it is necessary to consider video compression algorithms that are scalable, and can compress the same video input at various rates (consequently with different decoded output qualities). In other words, if the channel can provide different qualities of service (QoS) the video encoder should likewise be able to provide different rates and video qualities.This is possible thanks to the scalable nature of JPEG2000 called progressive transmission.

Finally, even if efficient channel coding is used, the variations in channel conditions due to roaming and fading will generally result in losses of information. Thus, the video application will have to provide sufficient built-in robustness to ensure that the quality of the decoded video is not overly affected by channel unreliability.

To address these challenges we define a framework utilizing new, JPEG2000 still image compression and RTP. The remainder of this paper is structured as follows.

In Section 3.2, we briefly describe the JPEG2000 with emphasise on tier-1 coding (block coding) and tier-2 coding (packeting). Section 3.3 overviews various types of built-in JPEG2000 error resilient coding and the techniques we adopted. The protocol communication structure of real-time JPEG2000 video streaming and the components of the system are presented in section 3.4.

## 3.2 JPEG2000 Codec Architectural Overview

The JPEG-2000 standard is comprised of numerous parts as shown in Figure 3.1. The codec is based on wavelet/subband [35] coding techniques and borrows heavily from ideas in the embedded block coding with optimized truncation (EBCOT) [52] scheme. Bit-plane coding techniques are used to code wavelet transform data. For entropy coding, a context-based adaptive binary arithmetic coder [56] is used—more specifically, the MQ coder from the JBIG2 standard [27].

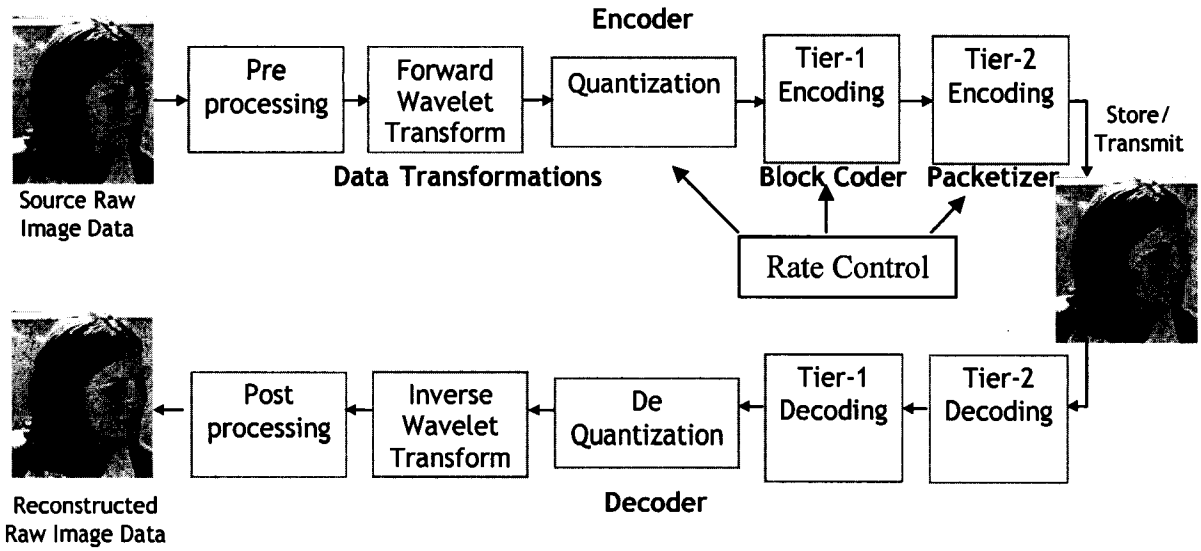



Figure 3.1: Block Diagram of the JPEG2000 System

For our purpose, the codec structure can be grouped into 3 categories with the emphasis on the last part. It is important to see that sometimes an image may be quite large in comparison to the amount of memory available to the codec. Consequently, it is not always feasible to code the entire image as an atomic unit. To address this issue, JPEG2000 allows tiling. The tiling feature refers to the partition of the original (source) image into rectangular non-overlapping blocks (tiles), which are compressed independently, as though they were entirely distinct images.

The main drawback to this approach is noticeable blocking artifacts known as tiling artifacts at tile boundaries, specifically in small images at low bit rates [53], since wireless imaging utilizes relatively small image sizes for handheld devices in the range of 160x120 pixel and with low bit rates. We are not deploying tiling in our application.

### 3.2.1 Data Transformations

The JPEG2000 standard specifies transformations that are applied to the image samples to produce quantized subband sample indices; the quantization indices are then passed to the block coder. The various transformations are identified in Figure 3.2.

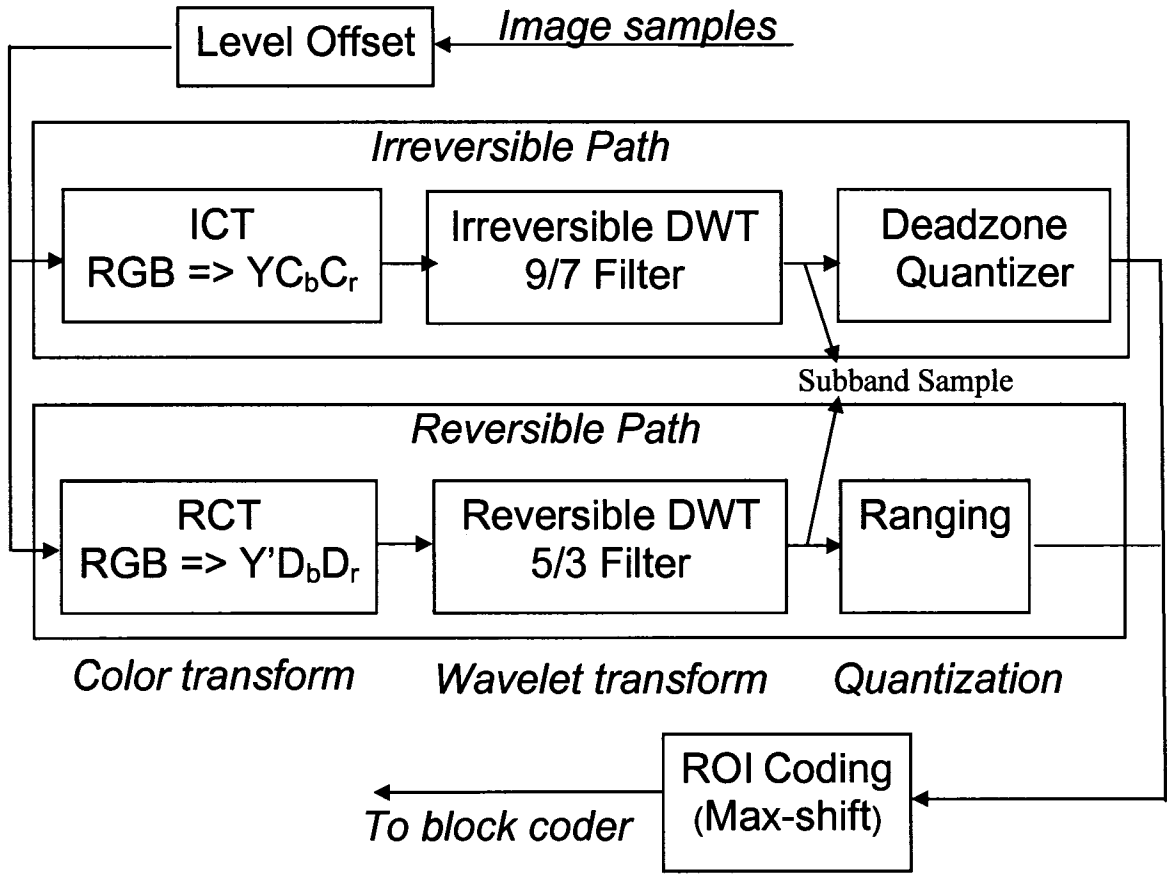



Figure 3.2: Block Diagram of the Data Transformation

The core operations in Figure 3.2 are those of the Discrete Wavelet Transform (DWT) and Deadzone Scalar Quantization (DZQ).The elements identified in the figure serve the following purposes:

**Level offset:** ensures that input sample data have a nominal dynamic range that is approximately centered about zero by subtracting a bias of $2^{B-1}$ from each sample value where B-bit image sample values are unsigned (non-negative) quantities, so that the samples X[n] have a signed representation in the range

$$-2^{B-1} \le X[n] < 2^{B-1}$$

**Color transform:** The color transform is optional. The transform converts the RGB data into an "opponent" color representation, with a luminance (or intensity) channel and two color difference channels (YCbCr). This has the effect of exploiting some of the redundancy between the original color components. There exist two color

transformations that take different paths; irreversible real-to-real in nature color transform (ICT), and reversible integer-to-integer color transform (RCT). The irreversible path is most easily described in terms of real-valued samples which are normalized (through division by $2^B$) to the "unit range"

$$-\frac{1}{2} \le X[n] \le \frac{1}{2}$$

**Discrete Wavelet Transform:** Figure 3.3 illustrates discrete wavelet transformation. Two specific wavelet transforms supported by the JPEG2000 codec are the 5/3 and 9/7 color transforms. The 5/3 transform [11] is reversible, integer-to-integer, nonlinear and can be used in either the lossy or lossless case. The 9/7 transform [4] is nonreversible and real-to-real,and is used in the case of lossy coding.

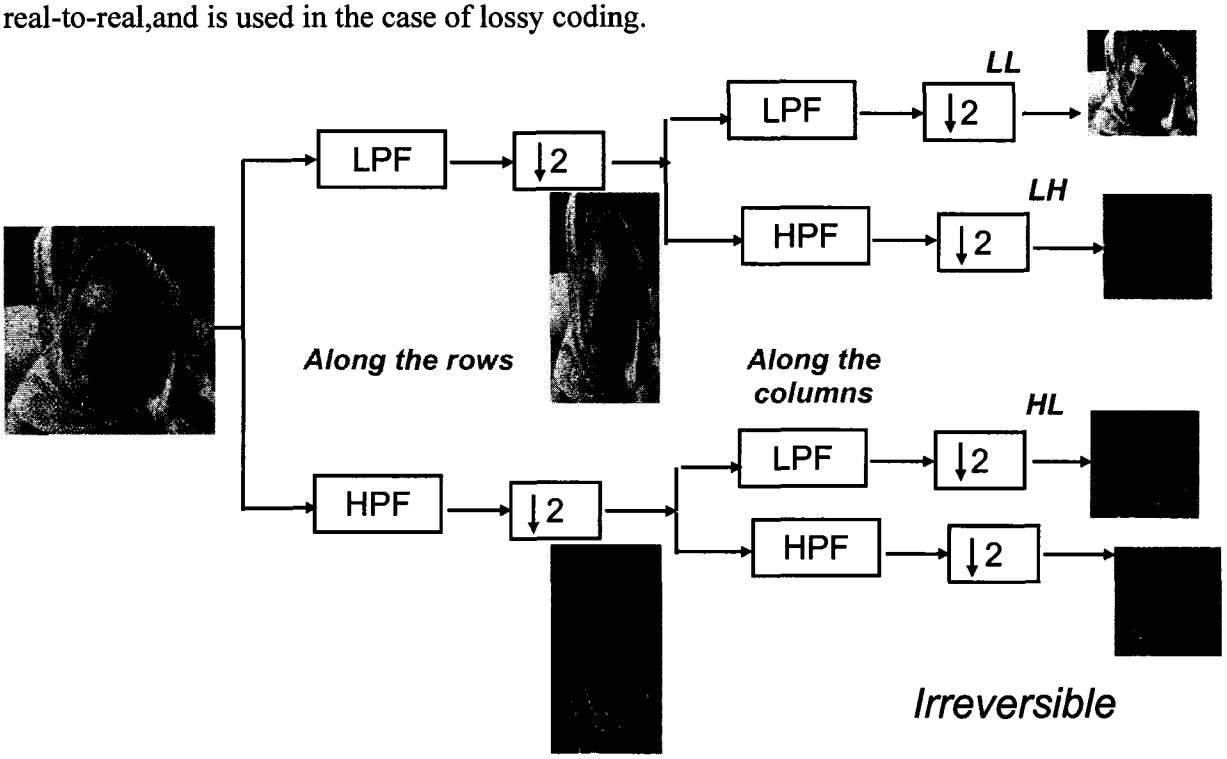



Figure 3.3: Wavelet Transform

**Division into resolutions and sub-bands:** Each image component is wavelet transformed with N decomposition levels. As a result, the component is available at N+1 distinct resolutions, denoted r = 0,1,...,N. The lowest resolution, r = 0, is represented by the NLL band.

In general, resolution r is obtained by discarding sub-bands nHH, nHL, nLH for n = 1 through N-r and reconstructing the image component from the remaining sub-bands. The coordinates of any particular sub-band, b, yields upper left hand sample coordinates $(bx_0, by_0)$ and lower right hand sample coordinates $(bx_1 - 1, by_1 - 1)$ where

$$bx_0 = \left\lceil \frac{cx_0 - (2^{n_b - 1}.xo_b)}{2^{n_b}} \right\rceil, \quad bx_1 = \left\lceil \frac{cx_1 - (2^{n_b - 1}.xo_b)}{2^{n_b}} \right\rceil$$

$$by_0 = \left\lceil \frac{cy_0 - (2^{n_b - 1}.yo_b)}{2^{n_b}} \right\rceil, \quad by_1 = \left\lceil \frac{cy_1 - (2^{n_b - 1}.yo_b)}{2^{n_b}} \right\rceil$$

Where $n_b$ is the decomposition level associated with sub-band b and the quantities $(xo_b, yo_b)$ are given as follows:

$$n_b(LL) = (0,0), \; n_b(HL) = (1,0), \; n_b(LH) = (0,1), \; n_b(HH) = (1,1)$$

Figure 3.4 illustrates a typical 3-level decomposition structure and its actual 3-level, discrete wavelet transform subbands for the monochrome image .

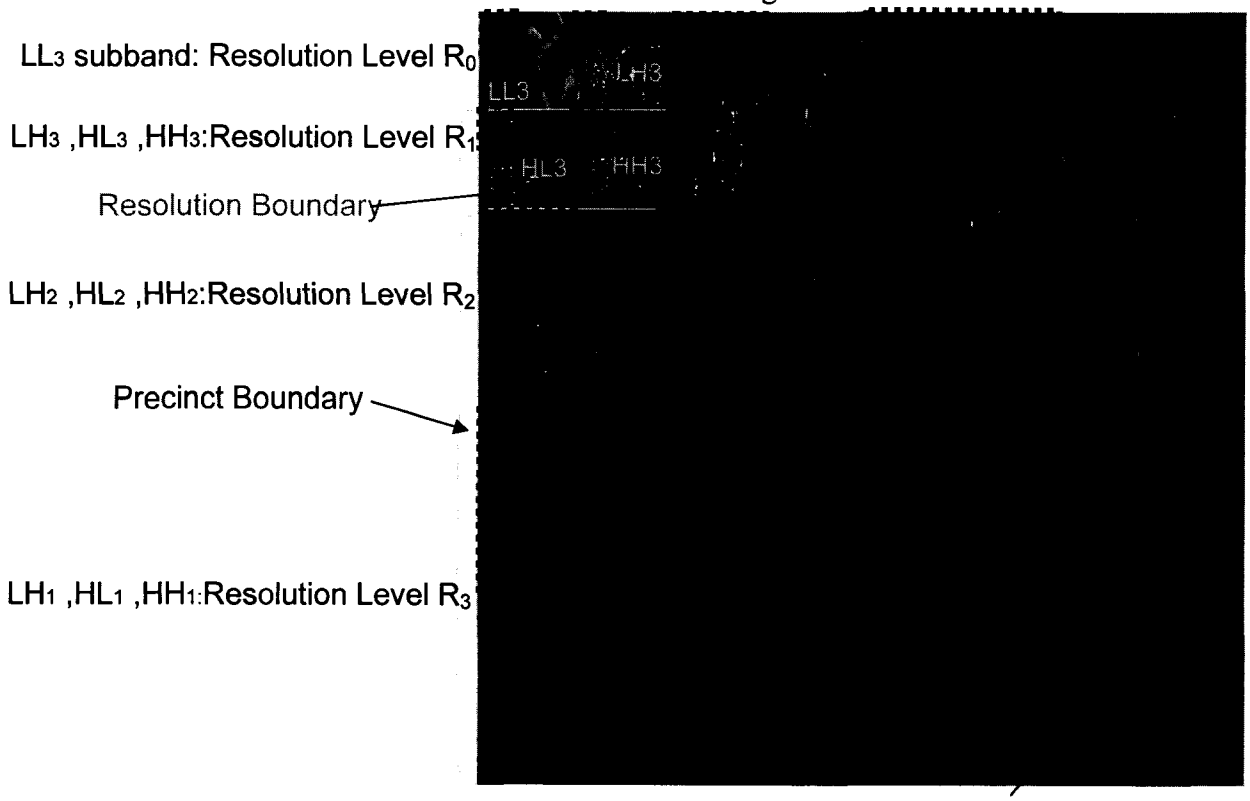



Figure 3.4: 3- Level Decomposition (N) Sub Band Structure

**Quantization-Ranging :** In the encoder, after the image data has been transformed, the resulting coefficients are quantized. The standard requires that sub-band samples be subjected to deadzone scalar quantization. The quantization operation is defined by the step size parameter, $\Delta_b$ through

$$q_b[n] = sign(y_b[n]) \left\lfloor \frac{|y_b[n]|}{\Delta_b} \right\rfloor$$

Where $y_b[n]$ denotes the samples of sub-band $b$, with a unit nominal range of $-\frac{1}{2} to \frac{1}{2}$, while $q_b[n]$ denotes their quantization indices.

In the reversible path (Ranging), the quantization indices supplied to the block coder are identical to the integer sub-band samples;

$$q_b[n] = y_b[n]$$

Quantization of transform coefficients is one of the two primary sources of information loss in the coding path, the other source being the discarding of coding pass data as will be described later. In the loss-less compression mode, the quantizer step size is always fixed to one, effectively bypassing quantization and forcing the quantizer indices and transform coefficients to be one and the same. In such a case, lossy compression is also possible but rate control is achieved by truncation of the embedded code-block bit-streams. It should be noted that the step sizes specified in the code-stream are relative and not absolute quantities. That is, the quantizer step size for each subband is specified relative to the nominal dynamic range of each subband's sample values.

### 3.2.2 Block Coder (Tier-1)

The data transfomation module outputs the quantizer indices of each sub-band to the block coder. In the tier-1 coding stage of the encoder the quantizer indices in each sub-band are partitioned into code-blocks, then each of these code-blocks is independently coded by making use of a bit-plane coding technique. There are three coding passes per bit plane where no interband dependencies are exploited. In order, these passes are as follows: 1) significance, 2) refinement, and 3) cleanup. All three types of coding passes

and code blocks scan the samples of a code block and the precincts' code blocks recpectively in the same fixed order shown in Figures 3.5 and Figure 3.6.

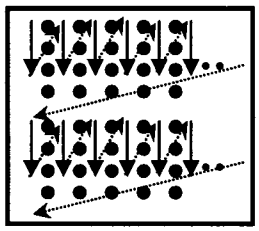


Figure 3.5: Sample Scan order within Code Block

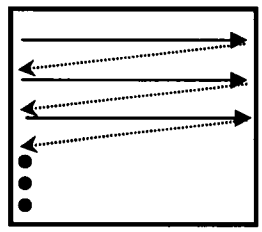


Figure 3.6: Code Block Scan order within Precinct

For each code-block an embedded code is produced, formed as a result of many coding passes. The first step of the block coder is the partitioning of each resolution into precincts that are further sub-divided into code-blocks, which are then coded independently. Code-blocks make use of a bit-plane coding alghorithm. Now, we introduce notation to formalize the Precinct and code blocks.

The precinct partition is anchored at location (0, 0), so that the upper left hand corner of any given precinct in the partition is located at integer multiples of $(2^{PPx}, 2^{PPy})$ where *PPx* and *PPy* are signaled in the COD or COC markers . *PPx* and *PPy* may be different for each component and resolution.

The number of precincts that span the image at resolution $r$ is given by:

$$Nunmber\,\mathrm{Pr}\,ecinctWide = \left\lceil \frac{rx_1}{2^{PP_x}} \right\rceil - \left\lceil \frac{rx_0}{2^{PPx}} \right\rceil \qquad Numbe\,\mathrm{Pr}\,ecinctHigh = \left\lceil \frac{ry_1}{2^{PP_y}} \right\rceil - \left\lceil \frac{ry_0}{2^{PPy}} \right\rceil$$

Each precinct P is identified by a pair of indices, $[p_1, p_2]$ which range over:

$$0 \le p_1 < N_1^{P,c,r}, \qquad 0 \le p_2 < N_2^{P,c,r}$$

Here, $N_1^{P,c,r}$ and $N_2^{P,c,r}$ identify the number of precinct rows and precinct columns respectively that are required in component $c$ at resolution $r$.

In JPEG2000, the sub-bands are further partitioned into rectangular code-blocks, which are then coded independently for the purposes of coefficient modeling and coding. The size of each element of the partition is determined from two parameters, *xcb* and *ycb*, which are signalled in the COD or COC markers and are the same for all sub-bands in the image at the same resolution $r$.

Specifically, the code-block size for the sub-bands is determined as $2^{xcb'}$ by $2^{ycb'}$ where

$$xcb' = \begin{cases} \min(xcb, PPx - 1), r > 0 \\ \min(xcb, PPx), r = 0 \end{cases} \qquad ycb' = \begin{cases} \min(ycb, PPy - 1), r > 0 \\ \min(ycb, PPy), r = 0 \end{cases}$$

These equations reflect the fact that the code-block size is constrained both by the precinct partition size and the nominal code-block size, the parameters of which *xcb* and *ycb*, are identical for all sub-bands in the image. Like the precinct partition, the code-block partition is anchored at (0,0). Thus, all first rows of code-blocks in the partition are located at $y = m2^{ycb'}$, and all first columns of code-blocks are located at $x = n2^{xcb'}$.

### 3.2.3 Packetizer (Tier-2)

In tier-2 encoding, the coding pass information is packaged into data units called packets, in a process referred to as packetization. A packet is nothing more than a collection of code blocks coding pass data. Each code-block's embedded bit-stream is distributed across a number of quality layers, $Q_l$, as depicted in Figure 3.5.

Mathematically, it can be written that $Z_i^l$ for the number of coding passes for code-block $Bi$ which may be decoded from quality layers $Q_0$ through $Q_l$. Code-block $B_i$ contributes a total of $L_i$ code bytes to these layers. Its incremental contribution to any particular layer, $Q_l$, consists of code bytes $L_i^{(z_i^l)} - L_i^{(z_i^{l-1})}$ and represents $Z_i^l - Z_i^{l-1}$ coding passes.

Since the information in packets belonging to the first layer (layer 0) is required to decode the information in the second layer (layer 1) and so forth, the sensitivity of data to corruption clearly decreases as we move from lower (layer 0) to higher (layer 5) quality layers. As a result, we would expect to find that a protection scheme that assigns stronger forward error correction (FEC) codes to lower layers than the higher layers can outperform a uniform protection strategy.

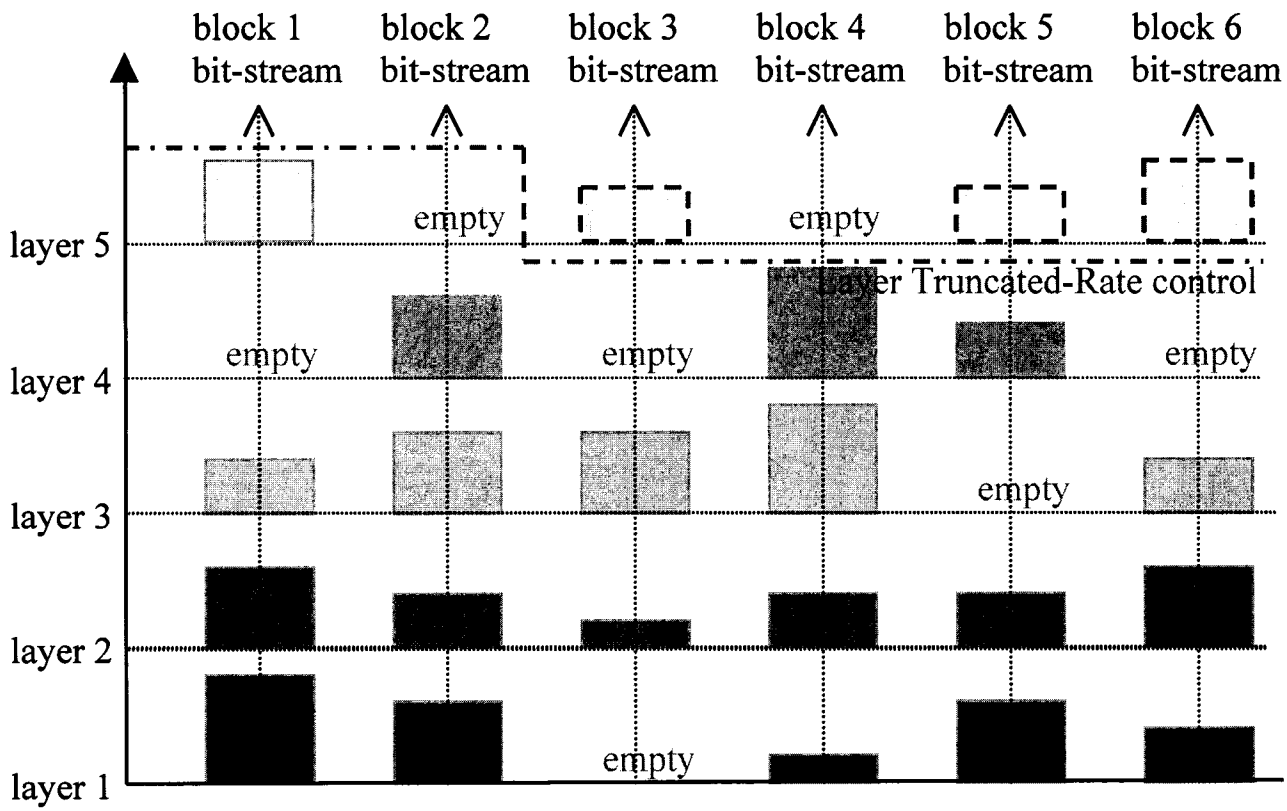



Figure 3.7: Layer Formation and Truncation

The resulting packet structure can be expressed as the following. Within each packet $P_{c,r,p,l}$, the code-block contributions are ordered in a deterministic fashion: first by sub-band (HL, LH then HH) within resolution level *r,* and then in raster scan fashion within precinct p of each sub-band. There is a nominal contribution from each code-block within the packet scope; that is, each code-block within the scope of precinct p in resolution level r of component c. However, any or all of these nominal contributions may be empty. Packet data is introduced by a packet header and followed by a packet body containing the actual code-bytes contributed by each of the relevant code-blocks. The order defined above is followed in constructing both the packet header and the packet body. The resulting packets are then output to the final code stream. A specific organization of the packet data in the output code stream, gives it features including rate scalability and progressive recovery.

**Progressive recovery:** Progression enables increasing quality, resolution, spatial extent, and/or color components as more bytes are decoded sequentially from the beginning of a compressed code-stream. The type of progression present in a JPEG2000 code-stream is

governed by the order in which packets appear within image-streams. There are five supported builtin progressions in the JPEG2000 standard, defined as: 1)resolution-layer-component-position 2)layer-resolution-component-position 3)resolution-position-component-layer 4)position-component-resolution-layer, and 5)component-position-resolution-layer. It is also possible to specify additional user-defined progressions at the expense of increased coding overhead.

**Final image Stream Structure:** The JPEG2000 code-stream structure begins with a main header. Each of these headers consists of a sequence of markers and marker segments. A marker is a two byte quantity of the form $FFXX_h$. A marker segment is made up of three parts: the type, length, and parameter fields. The type field identifies the nature of the marker segment. The length identifies the number of bytes in the marker segment, while the parameter field provides the segments information bearing parameters. The main header, as shown in Figure 3.6, begins with an SOC (start of code-stream) marker. This marker is followed immediately by the SIZ (image size) marker segment, which specifies global information such as image size, number of components, etc. There are two more required marker segments in the main header, in addition to several optional marker segments that are allowed to appear in the main header. These required and optional marker segments can appear in any order after the SIZ marker segment. The required marker segments are COD (coding style default) and QCD (quantization default), which provide default coding and quantization parameters, respectively. For example, the COD marker segment contains such information as the number of transform levels, the code block and precinct sizes, the progression order, and so forth, while the QCD marker segment contains quantization step size information. The optional marker segments include COC (coding style component) and QCC (quantization component), which provide the ability to override the default values specified in the COD and QCD marker segments on a component by component basis. The remaining optional marker segments include RGN (region of interest), which provides information for region of interest coding. The POC (progression order change) marker segment provides the ability to change progression orders within the code-stream. The PPM (packed packet headers: main header) marker segment can be used to move all packet headers to the main header. The PLM (packet lengths: main header) and TLM

(tile-part lengths: main header) marker segments can be used to record the lengths of compressed packets and tile-parts in the main header, respectively. These latter three markers may be useful for fast random access to the code stream.

**Packet Headers**: a packet consists of a packet header followed by a packet body. For the purpose of error resilient decoding, the code-stream syntax allows the optional inclusion of an SOP (start of packet) marker segment before some or all packets. Also, an EPH (end of packet header) marker can be included after each packet header.

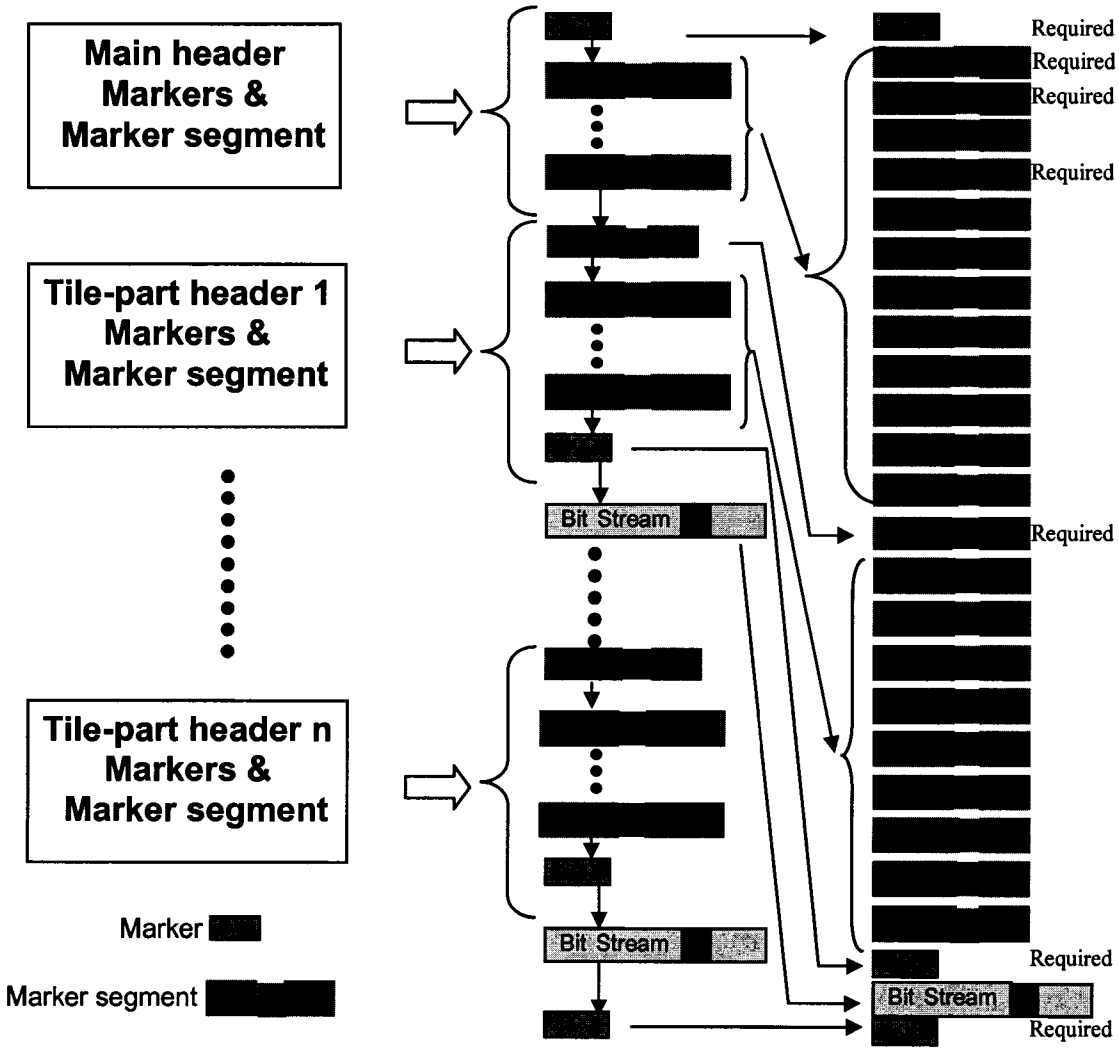



Figure 3.8: Construction of the Codestream

In addition to their usefulness for error resilient decoding, SOP marker segments and EPH markers can be useful for locating packets and/or packet headers. As a result of bit stuffing in the MQ coder, a $FF_h$ byte can never be followed by a byte having a value

greater than $8F_h$ within compressed pack-stream data. Thus, SOP = $FF91_h$ and EPH = $FF92_h$ can be located within the code-stream when present. Care must be exercised when searching for such markers however, because values within main, tile, and tile-part headers can take on any value, including values between $FF90_h$ and $FFFF_h$.

The main header contains crucial information such as image and code-block sizes, and is essential for the correct decompression of the code-stream. If sections of the main header are unavailable at the decoder, the decoder will not be able to decode the code-stream. Similarly, if a tile header is lost during transmission, the decoder might be unable to determine several parameters that will be required for the correct decompression of that tile. This might result in discarding the entire tile.

The main header and the tile header consist of a sequence of markers and marker segments. A typical marker segment is shown in Figure 3.6.

## 3.3 Built-in JPEG2000 Error Resilient Coding

Many applications require the delivery of image data over different types of communication channels. Typical wireless communication channels give rise to random and burst bit errors. Internet communications are prone to losses due to traffic congestion. To improve the performance of transmitting compressed images over error or loss prone transmission channels, error resilient bit stream syntax and tools are included in the JPEG2000 standard. These tools can be classified into packet and entropy level as shown in Table 3.1.

| Level Type | Description |
|---|---|
| Entropy | termination of the arithmetic coder for each coding pass<br>termination of the arithmetic coder for each segment<br>independently coding of code block<br>segmentation symbols"1010" |
| Packet | packet with resynchronization marker<br>shorter packets |

Table 3.1: Tools for Error Resilience

### 3.3.1 Entropy Coding Level

Partitioning the code-stream into independent code-blocks helps to isolate errors in one code block and prevent them from propagating through the entire code-stream. The decoder decodes the bit-stream for each block independently. Therefore, bit errors will not propagate from one block to the next as long as the block synchronization is maintained. To ensure block synchronization, we separate the blocks into different partitions, and put the bit-stream from different partitions into different packets. Using this approach, even if block synchronization is lost in one packet (because of errors in the packet header), we will be able to reestablish block synchronization in the next packet. Nevertheless a corrupted code-block bit-stream usually leads to objectionable artifacts in the decompressed image. Generally speaking, once an error occurs, the remainder of the embedded arithmetic and raw coded segments of the bit-stream is useless and subsequent decoding steps are likely to produce erroneous results.

To describe the error resilience process we need to make some assumptions. As in the tier-1 coding process, coding pass data can be encoded using one of two schemes (i.e. arithmetic or raw coding). Consecutive coding passes that employ the same encoding scheme constitute what is known as a segment. All of the coding passes in a segment can collectively form a single codeword, or each coding pass can form a separate codeword. Which of these is the case is determined by the termination mode in effect. Two termination modes are supported: per-pass termination and per-segment termination. In the first case, only the last coding pass of a segment is terminated. In the second case, all coding passes are terminated. Terminating all coding passes facilitates improved error resilience at the expense of decreased coding efficiency.

Basically JPEG2000 provides two error resilience modes at the entropy coding level to help minimize the impact of corruption in the packet data. One of these modes is the SEGMARK option that has been used to insert the special four symbol code, "1010", at the end of each cleanup pass and utilizes the per segment termination. A single error in the bit-plane is likely to corrupt at least one of the four "1010" symbols at the end of refinement pass. Upon detecting the corruption, an error resilient decoder should attempt to discard those coding passes that it suspects may contain errors. In the simplest case, the truncated bit-stream is then decoded over again from scratch. The result will be a

lower quality rendition of the relevant sub band samples, but less objectionable than the visual artifacts usually produced by decoding a corrupted bit-stream. Since the decoder has no way of knowing which of the three passes contained the error, it must discard them all.

JPEG2000 also offers an attractive alternative mode, which is to make use of the ERTERM and RESTART options to create a separate predictably terminated codeword segment for each coding pass in the case of coding pass termination. Any error in the bit-stream is likely to leave the coder in a state which is inconsistent with the predictable termination policy. An error resilient decoder may detect this condition at the end of the coding pass in which the error occurred. In this way, the decoder discards only those coding passes that are affected by the error.

Although the ERTERM and RESTART options provide superior error resilience to that offered by the SEGMARK mechanism, the overhead introduced by terminating each coding pass and explicitly signaling their lengths is larger than the cost of the four *SEGMARK* symbols per bit-plane.

Finally another factor contributing to the error resilience of a JPEG2000 code-stream is the number and spacing of quality layers. The layering concept imparts quality scalability to JPEG2000. By spacing the cumulative layer sizes more closely, each layer and each precinct packet represent a smaller incremental contribution to the image quality. The effect of impart quality is reflected in the distortion decrease in the SEGMARK mode.

### 3.3.2 Packet Level

Partitioning resolutions and sub-bands into multiple, smaller precincts can contribute to error resilience. Selecting smaller precincts yields smaller packet size, where a bit error is likely to result in less information loss (since, to some extent, bit errors in one packet do not affect the decoding of other packets), and thus reduce the overall error impact from packet loss or error. On the other hand, the use of smaller precincts results in a large number of packets being formed. This also results in an increase in the total amount of packet header information, which slightly degrades the overall compression efficiency. For this reason, one can expect to find that smaller precincts yield more robust code-streams.

Moreover, the inclusion of Packet resynchronization marker (packet header) within the data stream increases error resilience in some of the most demanding error-prone environments such as wireless channels. The packet headers appear in the code stream immediately preceding the packet data, unless one of the PPM or PPT marker segments has been used. If the PPM marker segment is used, all of the packet headers are relocated to the main header. If the PPM marker is not used, then a PPT marker segment may be used. In this case, all of the packet headers in that tile are relocated to the first tile-part header.

The packet header provides an indication to the receiver if some of the image packets are being lost or delivered out of order, and further enables synchronization between the encoder and the decoder during transmission. The code stream packets can be inserted into sections that correspond to the payload of the network packets and fill the network packets sequentially. The packet sequence number increases by one with each data packet sent and wraps around to zero when the maximum value is reached. The comprehensive details of this approach are discussed in Chapter 4.

## 3.4 JPEG2000 Video Streaming

Since the introduction of the RTP standard (RFC 1889) in 1996, it has been adopted by several standard organizations as the primary standard for real time multimedia transport over IP networks. To incorporate quality of service into the delivery of real time multimedia, it is essential to be aware of deliberately incomplete protocol specifications. These intentional deficiencies provide flexibility for application designers to develop robustness algorithms conformable to their needs. Our proposed RTP payload format is designed to deliver real time and robust packets of JPEG2000 video streams over a wireless network.

We first give a snapshot of RTP and incorporate it into the complete system framework of various other protocols and standards for real-time video transport on the Internet. Then describe the RTP format, which consists of the fix header and the payload header. Finally, we describe our suggested organization of JPEG2000 packets into the RTP packet for delivery.

### 3.4.1 RTP Snapshot

RTP comprises several profiles and payload formats respectively. It was developed by the Audio/Video Transport working group of the IETF and has since been adopted by the ITU as part of its H.323 series of recommendations.

Before we look in more detail at RTP and the design of systems using RTP, it will be useful to have an overview of the common structure of a system for the delivery of real-time multimedia utilizing RTP and the responsibilities of RTP senders and receivers in the system. Figure 3.10 depicts the block diagram of the sending and receiving process of RTP.

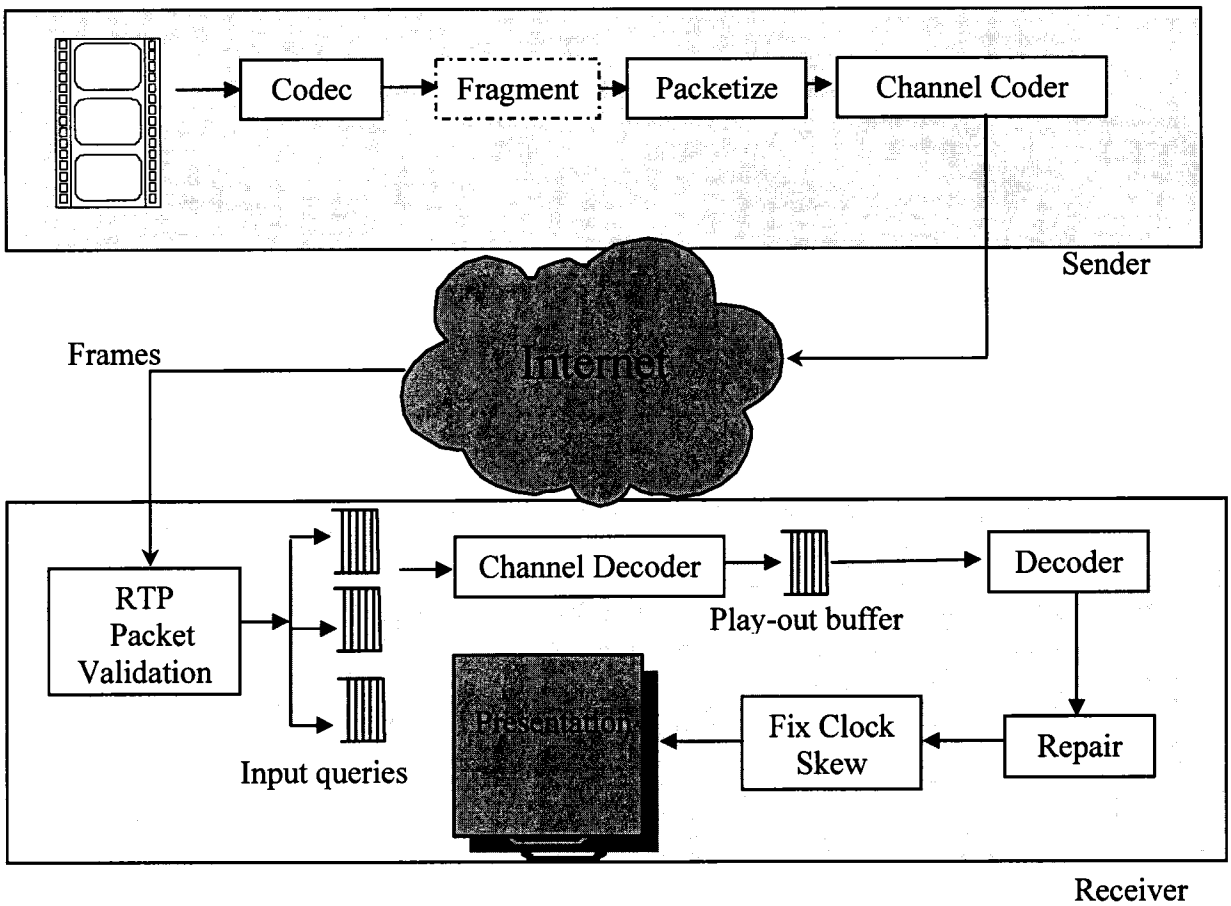



Figure 3.9: Block Diagram of an RTP Sender/Receiver

The RTP sender captures media data and produces compressed frames [fragments (large) frames or bundles several (small) frames] into the RTP packet and applies a robustness scheme before sending. Depending on the error correction scheme in use, a channel coder may be used to generate error correction packets or to reorder packets before

transmission. Finally, the sender produces periodic status reports and receives quality feedback from other participants to adapt its transmission.

On the client side, the RTP receiver collects packets, validates for correctness, puts them in sender-specified input queue, passes them to a channel-decoding routine to correct for loss, and puts in a play-out buffer (ordered by timestamps) to correct any reordering (additional packet buffering may be needed to remove any variation in inter-packet timing caused by the network). Upon frame completion when damaged or missing frames have been repaired, it decodes the frames and compensates for the clock skew to avoid gaps in the play-out. Depending on the codec used, it may be necessary to decode the media before missing frames can be repaired.

In addition to the RTP framework, a complete system will typically require the use of various other protocols and standards for real time multimedia transmission. Figure 3.11 illustrates a typical multimedia protocol stack for the purpose of real time video delivery.

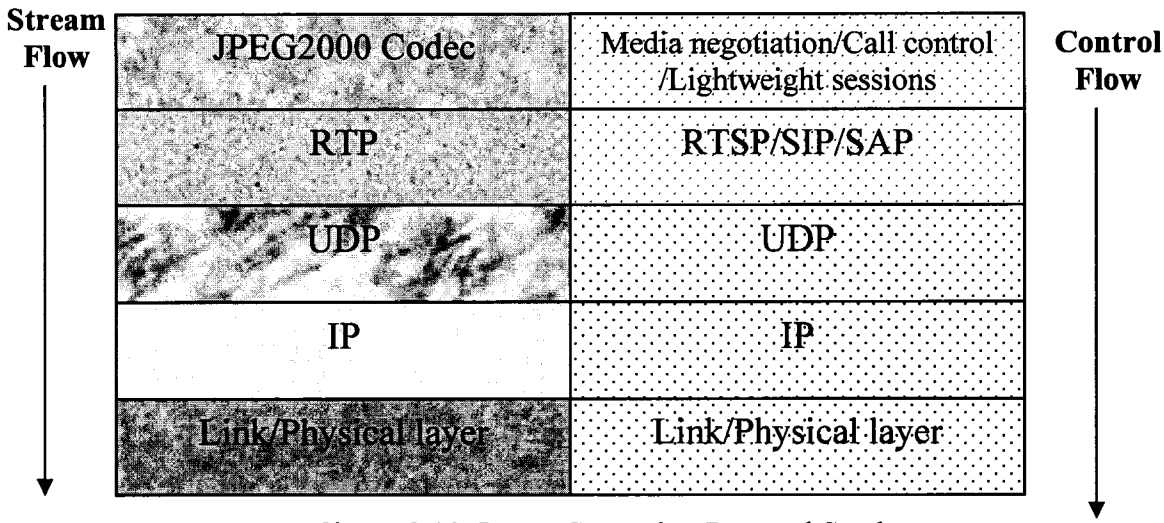



Figure 3.10: Image Streaming Protocol Stack

Most real-time multimedia applications on the Internet use the User Datagram Protocol, UDP [41], as their transport protocol. UDP compared to TCP has a low protocol processing overhead and no retransmission delays that make it attractive for delay sensitive applications.

As multimedia transmissions advance over the Internet, the partial checksum in MAC and UDP layers is proposed to prevent useful frames from being dropped and to reduce the number of retransmissions, thus reducing network load and delay. This comes from

the fact that real-time audio and video applications prefer errors in the payload to the loss of whole packets due to the capability of handling some errors in the payload by application codecs and lack of overall effect in outcome. In the UDP-Lite [33], the UDP checksum coverage is restricted to the most error sensitive part of the packet. In wireless networks, using a partial checksum to cover the sensitive parts (headers, at least) of the packet but not necessarily all user data should be mirrored also at the link or network interface level, where a stronger partial checksum can be deployed. However, the straightforward application of UDP-Lite to wireless networks is not possible because erroneous frames are dropped by the link layer (e.g. 802.11e [23]) before reaching the UDP layer.

To address this issue some papers suggest totally disabling the data integrity checks in the MAC layer [32]. Nevertheless, this is not a good solution for wireless networks since the Internet checksum used in UDP-Lite (and in UDP) was designed primarily for wired networks with far fewer errors.

Proposals [48] and [13] take a step forward in introducing the idea of reflecting the UDP-Lite policy of sensitive and non-sensitive data in the MAC protocol or Error Tolerant MAC. The payload data divided into A and B, the most and less class data respectively, where Class A that contains the data bits most sensitive to errors is covered by the CRC. The other bits are left unprotected, and thus packets with damaged Class B can be saved for the decoding process. In addition, the checksum extends coverage to the MAC, IP, UDP-Lite, and RTP headers. Packets with errors in the header are dropped, triggering retransmissions.

The results obtained by means of network simulations [47] for Error Tolerant MAC show that the assumption that bit corruption results in only minor distortion is valid only if the most sensitive part of the payload is check summed. The proposed technique also reduces the network load, since successful packet delivery requires, on average, fewer retransmissions.

### 3.4.2 RTP Packet Format

For each RTP packet in Figure 3.12, there are four parts to the packet:

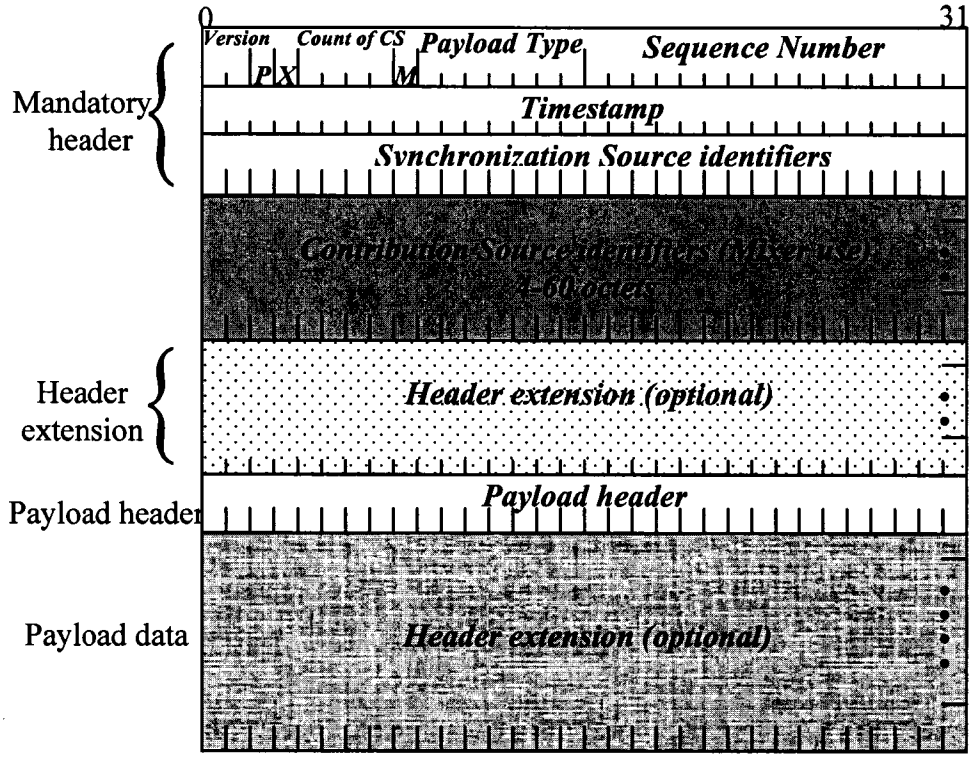



Figure 3.11: RTP Packet Format

- Mandatory RTP header
- Optional header extension
- Payload header (depending on the payload format used)
- Payload data itself

Here we discuss in more details the elements that offer error resilience to the bit stream.

*The mandatory RTP header*: is 12 octets in length and comprised of several elements. The payload type field of the RTP header identifies the media transported by an RTP packet to determine how to treat the data. A sequence number is used to identify packets, and to provide an indication to the receiver if packets are being lost or delivered out of order. The timestamp is derived from a media clock and denotes the sampling instant for the first octet of media data in a packet, and increases at a media-dependent rate (T) to the schedule the play out $(T+\tau)$ of the media data as shown in Figure 3.10. Finally, the synchronization source (SSRC) identifies participants within an RTP session.

If two packets contain data from the same sampling instant, they will have the same timestamp. Duplication of timestamps typically occurs when a large video frame is split into multiple RTP packets for transmission (the packets will have different sequence

numbers but the same timestamp Figure 3.11).

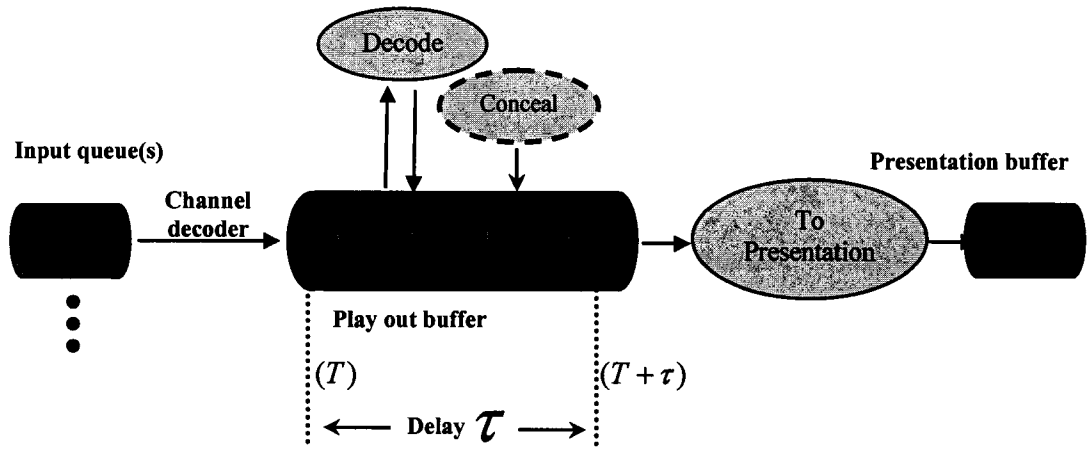



Figure 3.12: The Play Out Buffer

The primary use of the sequence number is loss detection. A gap in the sequence number space indicates to the receiver that it must take action to recover or conceal the missing data. The sequence number should always follow a continuous sequence, increasing by one for each packet sent, and never jumping forward or backward.

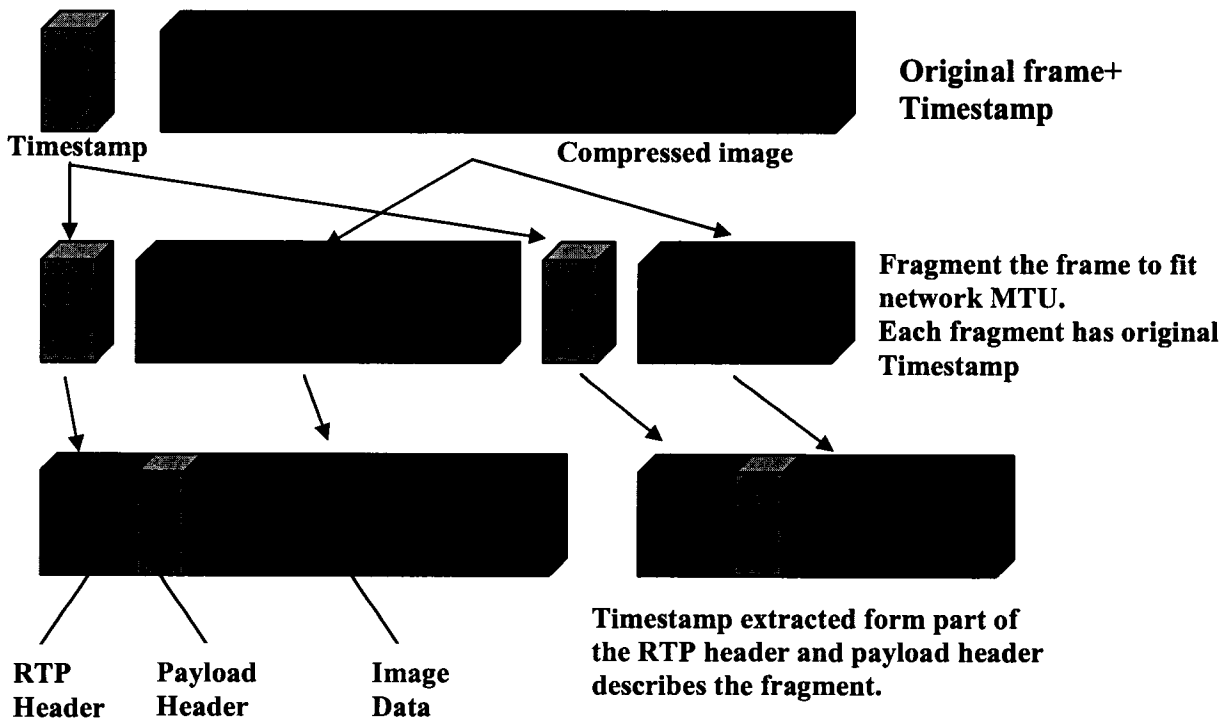



Figure 3.13: Fragmentation of the Image

*A payload header:* forms an additional header that is defined as part of the payload format specification that needs more information for optimal operation (e.g. error resilience). In fact, the primary reason for specifying payload headers is to provide error resilience for those formats that were not designed for use over lossy packet networks such as wireless networks.

Directly following any payload header, the final part of an RTP packet consists of one or more frames. Payload data size within an RTP packet is determined by two factors. First is the maximum transmission unit (MTU) of the network path. The MTU is the size of the largest packet that can be accommodated by a link. A common value is 1,500 octets, which is the largest packet an Ethernet can deliver. Secondly for long packets the latency is induced by the wait time to fill the packet. In other words, latency is related to the fact that a packet cannot be sent until the last octet of data it will contain is produced.

### 3.4.3 RTP packet format for JPEG 2000 payload

In our proposed scheme the RTP header fields that have a meaning specific to a JPEG 2000 video stream are defined as follows:

- **The mandatory JPEG2000 RTP header**

*Version Number (V):* We utilize the version number field as part of a packet validity check. Packets in which the version number is not equal to 2 (10) are invalid.

*Extensions(X):* header extensions are rarely used to provide for experiments that require more header information. Here X bit is set to zero.

*Count of contribution sources (CC):* In the case that multiple data sources may have contributed to an RTP data packet, the length of the Contribution Source identifiers list is indicated by the CC field in the RTP header.

*Marker (M):* This bit is used to mark events of interest within a media stream. It is set to one in the last RTP packet of the frame, as a hint that the application can begin decoding the frame.

*Payload type (PT):* It is dynamically assigned by means outside the scope of this Thesis.

*Timestamp:* The RTP for the JPEG2000 video stream timestamp is in units of 90 kHz as its clock rate. The initial value of the timestamp is randomly chosen intended to make known plain-text attacks on an encrypted RTP stream more difficult. The same value of timestamp should appear in the packets carrying the fragments of a frame.

- **JPEG2000 payload header**

As shown in Figure 3.12, we define JPEG2000 payload header as consisting of several parts, including the fixed header and any Contribution Source identifiers list and header extension to reassemble fragmented parts of an image and employ error resilience:

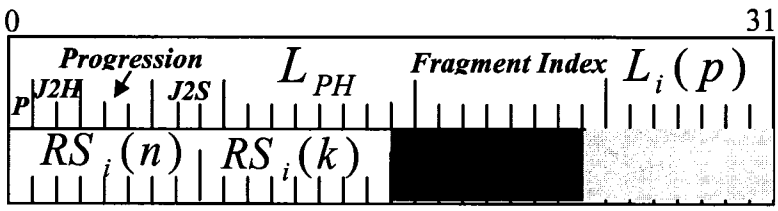



Figure 3.14: JPEG2000 Payload Header

*P (Packet type)*: 1 bit- indicates how a JPEG 2000 image is scanned fragmented. Generally this bit is set to 1 unless in certain condition such as direct point to point connection.

**0**- JPEG2000 image is arbitrary fragments and the timestamps value is the only source of indication in this mode.

**1**- JPEG2000 image is fragmented based on the payload header description

*J2H (JPEG2000 Header)*: 2 bits- indicates whether a JPEG2000 Header or piece of it is in the RTP packet. If there is no JPEG2000 Header, J2H has a value of 3. If there is just a part of a fragmented JPEG2000 Header, J2H has a value of 1. If there is the last part of a fragmented header, J2H has value of 2. If the whole JPEG2000 Header is inserted in the RTP packet, J2H has a value of 0.

| J2H | Description |
|---|---|
| 00 | whole JPEG2000 Header in the payload |
| 01 | piece of fragmented JPEG2000 Header in the payload |
| 10 | last part of a fragmented JPEG2000 Header in the payload |
| 11 | no JPEG2000 Header in the payload |

Table 3.2: JPEG2000 Header Indicator

*Progression:* 3 bits- signals the type of progression used in a JPEG2000 image and its packet respectively. We assign 000 to 101 to the 5 supported JPEG2000 progressions and the rest of the numbers to the user defined progressions.

*J2S (JPEG2000 Stream)*: 3 bits- J2S implies various combination of packet stream in the RTP packet. Several formats can be describe based on image size and targeted application.

$L_{PH}(PayloadHeaderLenght)$: 8 bits- assigns the length of payload header in bytes, not including previous fields.

$L_i(P)$: 7 bits- indicates the length of the $i^{th}$ JPEG2000 packet in the RTP packet.

$RS_i(n)$: 8 bits- represents the number of symbols per block (e.g. for 8 bits symbol n = 28−1 = 255) for the $i^{th}$ packet in the code stream.

$RS_i(k)$: 8 bits- represents the number k, k < n, of data symbols in the block for $i^{th}$ packet in the code stream.

# Chapter 4

# Robust Real Time Transmission of JPEG 2000 Packets

## 4.1. Introduction

Nowadays, the Internet's infrastructure is experiencing the rapid growth of wireless multimedia applications, which has resulted in huge investments in R&D to build coding efficiency and error resilience into multimedia and ensure improvements in the speed and quality of multimedia in error prone wireless environments. In this respect, the remarkable features of JPEG2000 make it a leading contender for wireless multimedia applications. An analysis of the performance and comparison study in [14] concludes that Motion JPEG2000 is very well suited for wireless applications.

Although JPEG2000 offers a set of tools for error resiliency, it does not guarantee an error free received image since it cannot correct transmission errors and does not address the presence of errors in the image header, even though it is the most important part of the code stream. Moreover the degree of protection varies depend on the application and its requirement. Thus the JPEG2000 standard has chosen to minimize the standardization of the error resilience tools. This approach leaves the implementers free to choose the best way to exploit the error resilience tools.

In this chapter we propose schemes for achieving robust JPEG2000 wireless imaging and discuss the challenges that will have to be overcome for these systems to become reality. We focus on image error resilience issues and outline how the requirements of robustness to communication channel conditions are being addressed in our system. We emphasize in particular the potential benefits of designing image transmission algorithms that are unequally protected and can adjust their performance in real time to increase the overall robustness of the system.

## 4.2. Unequal Packet Loss/Error Protection

The set of available error resilient tools in JPEG2000 standard works at both the entropy level and the packet level. The former are called built-in JPEG2000 error resilient tools and were already described and configured for the optimal performance in Chapter 3. These tools are predetermined and implemented by the standard specification where the users can enable/disable the tool. In order to achieve the optimal performance a trade-off between coding efficiency and redundancy is necessary. On the other hand; the packet level tools are open to the implementation of the techniques to satisfy the need of the application. In this respect, Packet Loss/Error Protection is an example of the user implemented scheme.

It is obvious that extending an equal level of protection to different parts of an image improves overall image quality; however, in case of limited bandwidth such as wireless channel and highly scalable code stream such as JPEG2000 it is not a wise choice to uniformly impose redundancy to all parts. Instead an effective technique is the use of UEP (Unequal Error Protection) which the encoder applies to the protection to different parts of the data unequally base on the importance and sensitivity of the data to errors.

### 4.2.1. UEP/ULP Schemes Review

A variety of techniques utilizing the UEP/ULP scheme have been proposed that define how to partition the data and which protection codes in what extent should apply. The popular codes are the Reed-Solomon [50] which is a special case of BCH [9] which is a multilevel, cyclic, and error-correcting, variable-length digital code.

In [30], the Reed Solomon codes are applied in order to provide unequal error protection (UEP) to the JPEG 2000 bit stream. In this paper, the impact of the existing error resilience tools offered by JPEG2000, including error concealment, resynchronize markers, layering and precincts has been examined. These tools are of substantial benefit, while dividing the image resolutions into multiple precincts is advantageous for error resilience.

The availability of multiple quality layers is of little benefit unless forward error correction codes are introduced to protect the layers unequally. Using a family of RS

block codes having codeword sizes of 15 nibbles (60 bits), it has been found that JPEG2000 code-streams can be made substantially resilient to errors.

Unequal protection of the quality layers is definitely beneficial and allows a code-stream to be efficiently protected over a range of different channel error conditions. Somewhat counter-intuitively, it has not found the use of multiple precincts to be of any benefit when RS codes are used to protect the compressed data either equally or unequally across quality layers.

It is worth noting that JPEG2000's error resilient built-in tools only detect the occurrence of errors, conceal the erroneous data and resynchronize the decoder. In particular, they do not correct transmission errors and do not address the occurrence of errors in the image header even though it is the most important part of the code stream. For these reasons, they are not sufficient in wireless imaging.

There exist some tools to protect the header of the image to prevent decoder crash in [40], where a backward compatible header error protection mechanism is described. It consists of the addition of a dedicated marker segment to a JPEG 2000 Header code stream that will contain the error correction data generated by a block error correction code (e.g. a Reed Solomon code). This mechanism leaves the original data intact, hence providing backward compatibility with the already standardized JPEG 2000. Neither side information from a higher level nor extra signaling encapsulation is needed, as the required information is directly embedded in the code stream and is also protected. Also, [18] and [42] proposed some algorithms to protect sensitive parts of the data including the header.

In an ARQ scheme in a mobile environment, error sensitivity information can be exploited by a proxy server at the base station to optimize the retransmission management and the protect data header. In particular, it has been found that the strategy that does not constrain the number of retransmission attempts for each packet, but rather for each frame, provides improved performance. This is mainly due to the fact that this strategy privileges the retransmission of the first packets of the code stream, achieving a good trade-off between channel rate and delivered quality.

Interleaving / de-interleaving is another solution proposed by [16]. Here the combined use of Reed-Solomon in UEP architecture along with interleaving improves performance in a variety of applications

One of the effective algorithms introduced by Moher et al. in [38] yield good results for memory less and fading channels, as it proposes a scheme exploiting RS codes and applying FEC unequally. The technique is based on priority encoding transmission PET [2] where apply to mpeg sequence in [34].

Also in a UEP solution, convolutional [20] codes are very useful for bit-error channels, and exhibit nice low-complexity features at both the encoder and decoder side. Turbo-codes [6] have been recently gaining in popularity due to their excellent error correction performance; in [5] turbo-codes have also been successfully applied to JPEG 2000. Convolutional codes and turbo-codes can be made rate-adaptive by means of puncturing the output stream [19], [55], thus allowing us to carry out unequal error protection by using a single punctured code. This allows the transmission of incremental redundancy in UEP ARQ/FEC schemes and continuous rate variation to change from low to high error protection within a data frame.

### 4.2.2. Proposed UEP Scheme

For the robust transmission of bit stream data two types of models exist. One is bit error loss models where random or burst error occurs in specific bit error rates (BER); the other is the packet loss model, which assumes the partition of data into packets with header and body. The effect of the errors in a packet is limited to the packet. The detection and correction of errors in each packet are independently investigated, and errors in one packet do not disrupt an ongoing reception of subsequent data packets.

Eventually, there are two models to develop an unequal error protection for image transmission. The first is the bit level loss model and the second is the packet level loss model. The packet loss models imply the undisrupted receipt of packets in which upon detection of an error in a packet either receiver requests the sender to retransmit the packet. The procedure continue until the packet is received free of error or the packet data is discarded in case the retransmission threshold is exceeded. The bit error model assumes the occurrence of errors at some random bits, where it implies a bit error rate

(BER). The mentioned models define the PLR (Packet Loss Rate) and BER that are frequently use in objective image qualification. For my work I purposefully pay attention to both models in order to improve code error resiliency in both the bit and packet level.

To develop the error resilient model, I look at the existing design structures for UEP codes. Most of the time the code resiliency is provided by an external coding scheme, such as the unequal error protection solutions proposed in [39]-[44].

The drawbacks of external schemes are well known; in particular, the error correcting scheme must either have full knowledge of the bit stream's syntax (resulting in the necessity to transmit extra information between the two coders), or protect it partially blindly (i.e. less efficiently, for instance by using statistical ratios to determine the various bit stream parts). Moreover, such schemes cannot take advantage of the various source coder functionalities such as the reordering of headers.

Compared to these external schemes, my scheme can be directly supplied by an embedded mechanism in the JPEG2000 syntax. As a consequence, it is proposed in this thesis to develop a scheme that embeds the protection within the JPEG2000 stream.

## 4.3. Data Partitioning and Interleaving

One of the advantages of JPEG2000 over the other compression methods is tiling, which refers to the partitioning of the original (source) image into rectangular, non-overlapping blocks (tiles) that are compressed independently, as though they were entirely distinct images [51]. The tiling feature might have been exploited to isolate the error in each tile of the image; however tiling affects the image quality both subjectively and objectively. Smaller tiles create more tiling artifacts compared to larger tiles and no-tiling (PSNR values are the average over all components). In other words, no-tiling and larger tiles perform visually better than smaller tiles. Image degradation is more severe in the case of low bit rates than in high bit rates. It is seen, for example, that at 0.125 bpp there is a quality difference of more than 4.5 dB between no-tiling and tiling at 64 × 64, while at 0.5 bpp this difference is reduced to approximately 1.5 dB.

In my design, the whole image is regarded as one tile and no tiling producer is chosen. For our purpose I selected an image of 180X120 pixels in height and width respectively.

The image size is in most cases the standard dimension for the image exploited in handheld devices.

In the sections that follow, I describe the packet coding process in more detail. For ease of understanding, I have chosen to explain this process from the point of view of the encoder. The decoder algorithms, however, can be trivially deduced from those of the encoder.

In the encoder, tier-1 encoding is followed by tier-2 encoding. In tier-2 encoding, the coding pass information is packaged into data units called packets, in a process referred to as packetization [1]. The packetization process imposes a particular organization on coding pass data in the output code stream [10]. Each packet is comprised of two parts: a header and a body. A packet is nothing more than a collection of coding pass data. The header indicates which coding passes are included in the packet, while the body contains the actual coding pass data itself.

More than one ordering of packets in the code stream is supported. Such orderings are called progressions. There are five built-in progressions that are defined: 1) layer-resolution-component-position ordering, 2) resolution-layer-component-position ordering, 3) resolution-position-component-layer ordering, 4) position-component-resolution-layer ordering, and 5) component-position-resolution-layer ordering.

### 4.3.1. Proposed Progressive Scheme

One solution in [42] to achieve UEP through RCPC codes is progressive recovery by fidelity scenario, which is implemented by considering one noise sensitive class for each quality layer, because the first quality layers are more influential in terms of error sensitivity.

In summary, it proposed the following classification for a JPEG2000 bit stream ordered in Nc quality layers in a progressive quality transmission context:

- Class 0: Main header and Tile-part header.
- Class 1: Packets for Quality layer 1.

  ...
- Class Nc: Packets for Quality layer Nc.

The lower classes have more protection overhead.

My scheme achieves UEP through CPRL progression (Component-Position-Resolution-Layer Progression).

*for each c*

*for each p*

*for each r*

*for each l*

*include Packet* $_{c,r,p,l}$

This progression is primarily "progressive by component," as all packets from component 0 precede all packets from component 1, etc.

As shown in Figure 4.1, my progression order has two advantages. In particular, upon the receipt of a corrupt packet or detection of a lost packet, the decoder is resynchronize pointer can jump to the preceding resolution layer (next highest resolution). In this scenario, I quickly resynchronize decoder with the receiving data stream on fly and discard the data stream, which increases the distortion of the image.

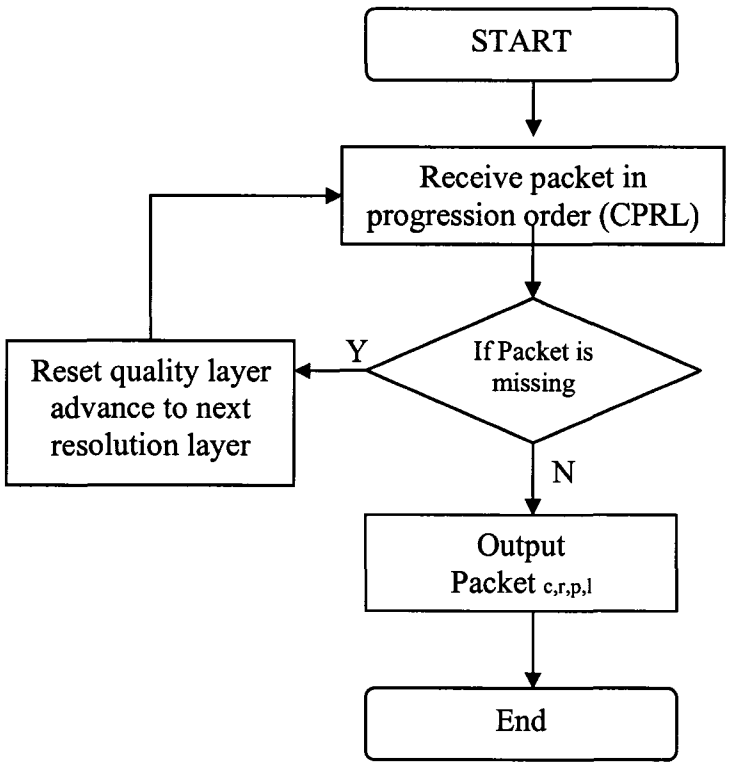



Figure 4.1: Packet Corrupted Detection Algorithm

## 4.3.2. Packetization

As described above, in tier-2 coding, the coding pass information is packaged into data units called packets, in a process also referred to as packetization. Each packet consists of a packet header followed by a packet body. For the purpose of error resilient decoding, the code-stream syntax allows the optional inclusion of an SOP (start of packet) marker segment before some or all packets. Also, an EPH (end of packet header) marker can be included after each packet header.

The packet headers will be in only one of three places within the code stream:

- In PPM marker segments
  - If the PPM marker segment is present, all the packet headers are found in that marker segment. In this case, the PPT marker segment and packets distributed in the bit stream of the tile-parts are disallowed.
  - There may be multiple PPM marker segments in the main header.
- In PPT marker segments
  - If there is no PPM marker segment then the packets can be distributed either in a PPT marker segment in the first tile-part (Tsot = 0)
- In the code stream

I chose the first option for reasons that will appear later in this section.

In order to improve the resiliency of my code I decrease the size of each packet which results in an increase in the number of packets. The size of the packets will be defined indirectly by precinct size as follows:

Consider a particular component and resolution whose bounding sample coordinates in the reduced resolution image domain are (trx0, try0) and (trx1-1, try1-1), as already described. Figure 4.2 shows the partitioning of this tile component resolution into precincts. The precinct partition is anchored at location (0, 0), so that the upper left hand corner of any given precinct in the partition is located at integer multiples of $(2^{PP_x}, 2^{PP_y})$ where PPx and PPy are signaled in the COD or COC markers. PPx and PPy may be different for each component and resolution.

The number of precincts that span the tile-component at resolution $r$ is given by

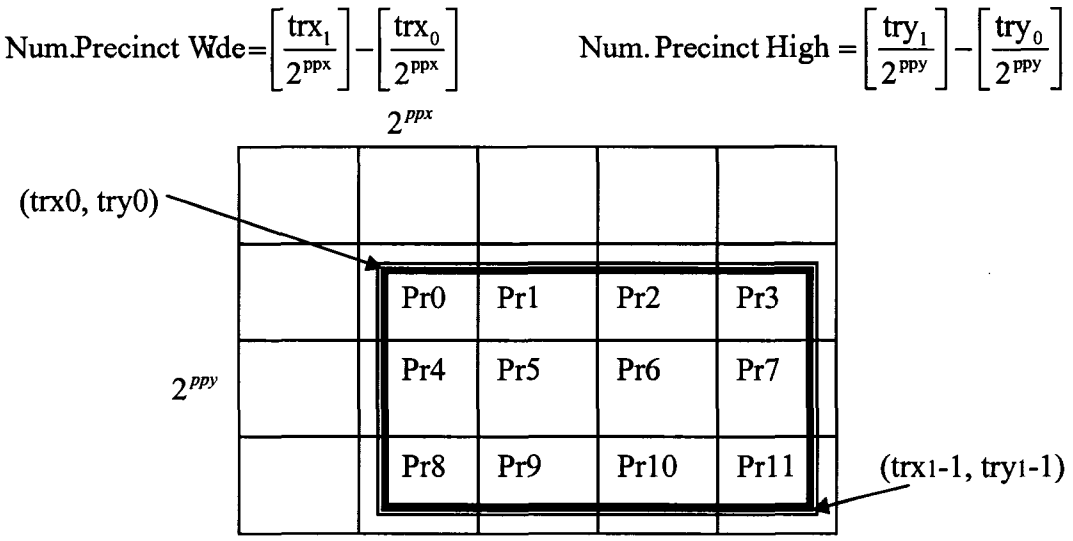



Figure 4.2: Precincts of One Resolution

As indicated above the packet header can be optionally located in the main header. To extend the error resilience to headers and avoid (or strongly limit) decoding crashes due to the presence of errors in the headers, I define an error protection mechanism for protecting the Main Header. It is to be noted that this protection is extended to the packet headers that are relocated within the Main header.

## 4.3.3. Packet validation algorithm

Upon the receipt of RTP packets, these packets will be validated and extracted from their input queue and inserted into a play out buffer for presentation, sorted by their RTP timestamps $T_s(i)$.

Frames are held in the play out buffer for a period of time $(\tau)$, called the delay time, to smooth out timing variations caused by the network. Also holding the data in a play out buffer allows the pieces of fragmented frames to be received and grouped, and allows any error correction data from the channel decoder to arrive. The frames are then decompressed, any remaining errors are concealed, and the media is rendered for the user. Figure 3.10 in the previous chapter illustrates the play out process.

The play out buffer comprises a time-ordered, linked list of nodes or RTP packets. Each RTP packet represents part of the image data, whole image data, or a number of images, with associated timing information. The data structure for each RTP packet contains

pointers to the adjacent RTP packets, the arrival time, RTP timestamp $T_s(i)$, and desired play out time $T+\tau$ for the image(s), and pointers to both the compressed fragments of the image (the data received in the RTP packets) and the uncompressed media data. Figure 4.3 illustrates the RTP packet and fragmented part of the data structures.

In my application, the timestamps of the fragmented RTP packet along with the relevant fields of the payload header described in Section 3.4.3 construct the RTP packet where the size of the image exceeds MTU.

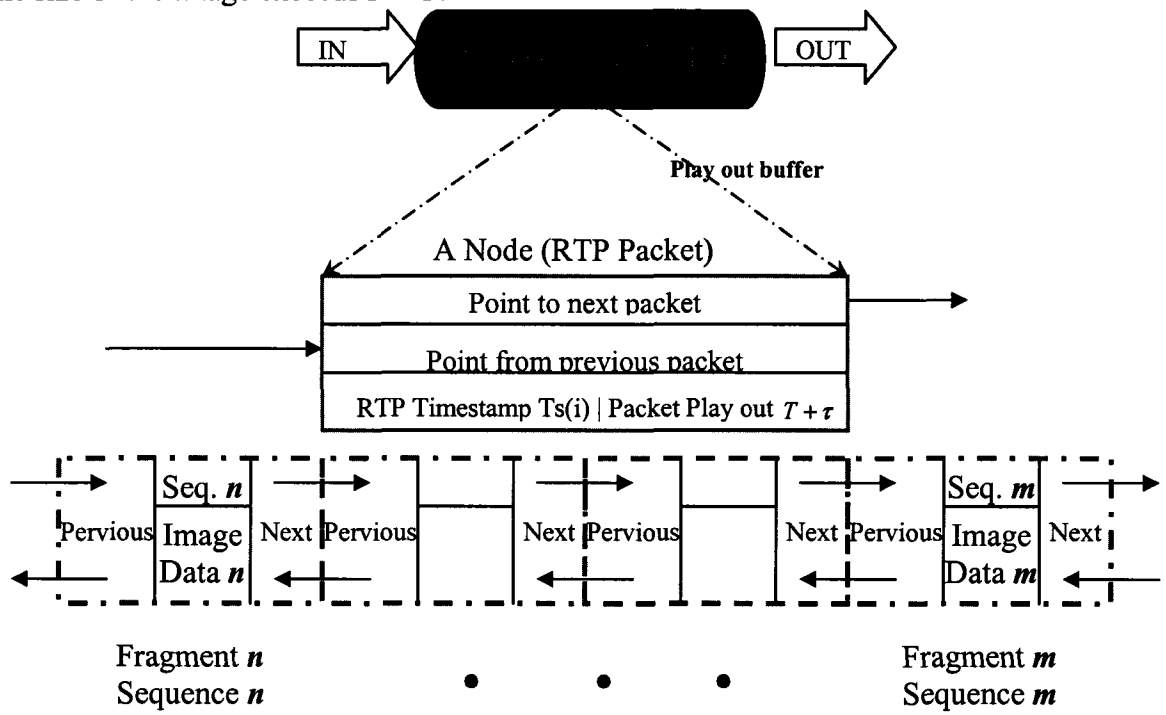



Figure 4.3: RTP Packet and Fragment Structure

In this scheme, the arrived play out buffer is created by a linked list of RTP packets and the fragmented image, where the compressed image of the newly arrived packet is linked from the play out buffer node for later decoding.

The newly created node resides in the play out buffer until its play out time is reached. During this waiting period, packets containing other fragments of the frame may arrive and are linked from the node. Once it has been determined that all the fragments of a frame have been received by means of payload header fields, the decoder is invoked and the resulting uncompressed frame is linked from the play out buffer node. Determining that a complete frame or image has been received depends on the payload header:

Based on the MTU limitation, The RTP sender often generates multiple RTP packets per video frame or image, with the RTP J2S marker bits being set to indicate the RTP packet

containing the last fragment of the RTP packet. However receiving an RTP packet with the J2S marker bits set to indicate last packet fragment does not necessarily mean that the complete frame has been received, since packets may be lost or reordered in transit. Instead, it gives the highest RTP sequence number for an image.

Once all RTP packets with the same timestamp but lower sequence numbers have been received, the image is complete. Whether the image is complete can be easily determined if the packet with the J2S marker bits for the previous image was received. If that packet was lost, as revealed by a timestamp change that appears without the marker bit, and if only one packet is lost according to the sequence numbers, then the first packet after the loss is the first packet of the image. The decision of when to invoke the decoder depends on the receiver and is not specified by RTP. Images can be decoded as soon as they arrive or kept compressed until the last possible moment. The choice depends on the relative availability of processing cycles and storage space for uncompressed images, and perhaps on the receiver's estimate of future resource availability.

The trade-off in play out buffer operation is between fidelity and delay: An application must decide the maximum play out delay it can accept, and this in turn determines the fraction of packets that arrive in the time to be played out. A system designed for interactive use—for example, video conferencing or telephony—must try to keep the play out delay as small as possible because it cannot afford the latency incurred by the buffering. Studies of human perception point to a limit in round-trip time of about 300 milliseconds as the maximum tolerable for interactive use; this limit implies an end-to-end delay of only 150 milliseconds including network transit time and buffering delay if the system is symmetric. However, a no interactive system, such as streaming video, television, or radio, may allow the play out buffer to grow up to several seconds, thereby enabling non interactive systems to handle variations in packet arrival times better.

While buffering the packets for play out, the validation algorithm examines the received packets for any out of order, lost or delayed packets. The packet ordering and loss validation algorithm depicted in Figure 4.4 guarantees the ordering of the packets in the play out buffer. To establish a connection between the client and server, first we initiate the sequence number of the receiver based on the sender's first packet sequence number. According to my algorithm, the initial packet sequence number is first compared to the

buffered sequence numbers. If the sequence numbers are equal, we output the previous play out buffered image instead of the incoming packet. Afterward we increment the initial sequence number of the receiver and repeat the process.

Now suppose that the sequence numbers are not the same. Thus the packet is extracted from input queue and its sequence number is compared to the sequence number of the receiver. If they are the same, the outcome is the input queue packet; otherwise we conclude that the arrived packet is out of order or that the expected packet is lost. We check whether the maximum out of order limit has been reached. Packets with sequence number exceeding the limit are discarded, and the rest is buffered for future use.

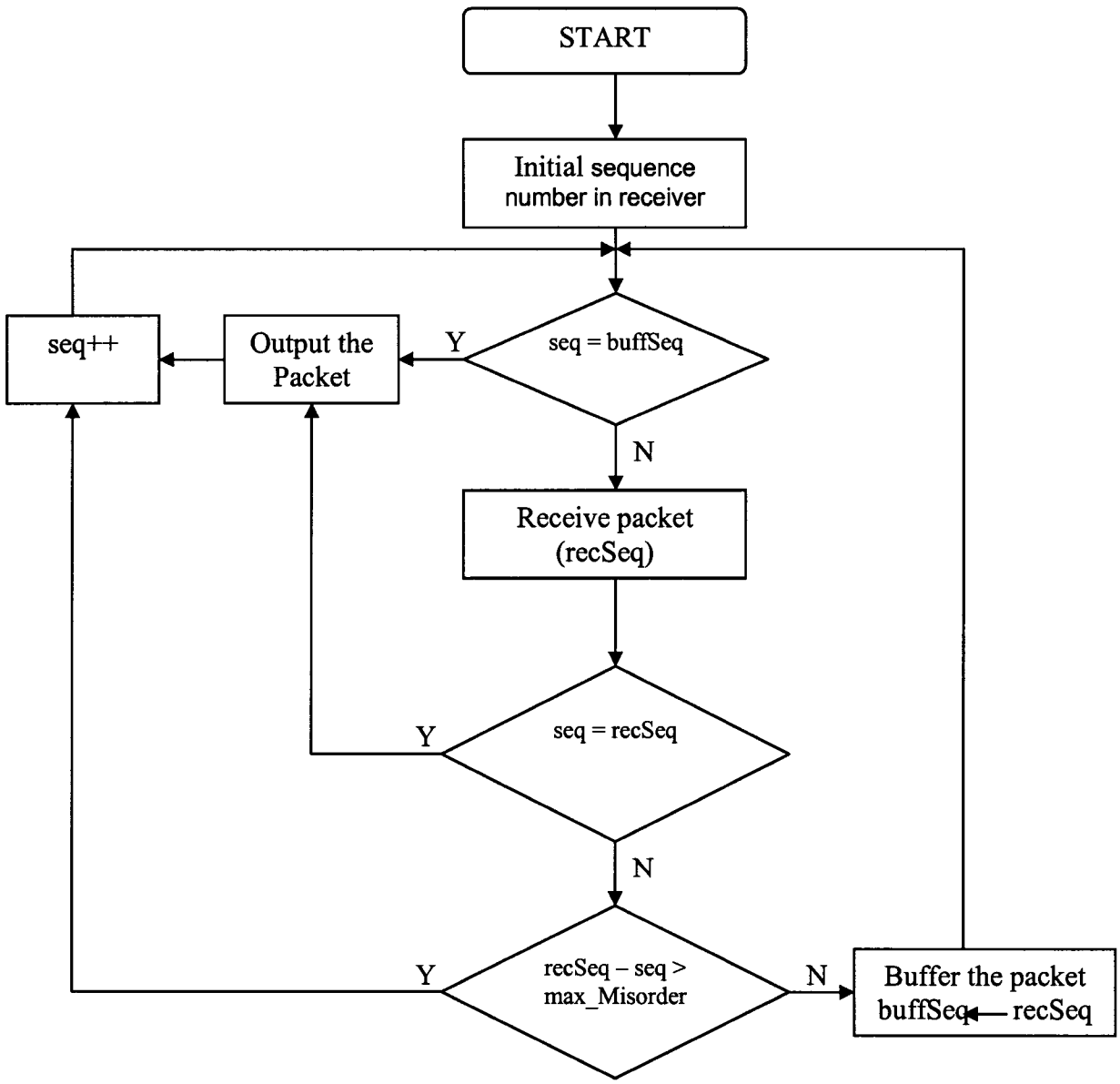



Figure 4.4: Packet Ordering and Loss Validation Algorithm

## 4.4. Error resilience Code Stream Syntax

My proposal is error resilient at the packet level where I packetize the code stream and insert the packet header (PPM) and EPH marker into the main header and packet resynchronization marker (SOP) in the code stream. Figure 4.5 illustrates the final code stream syntax.

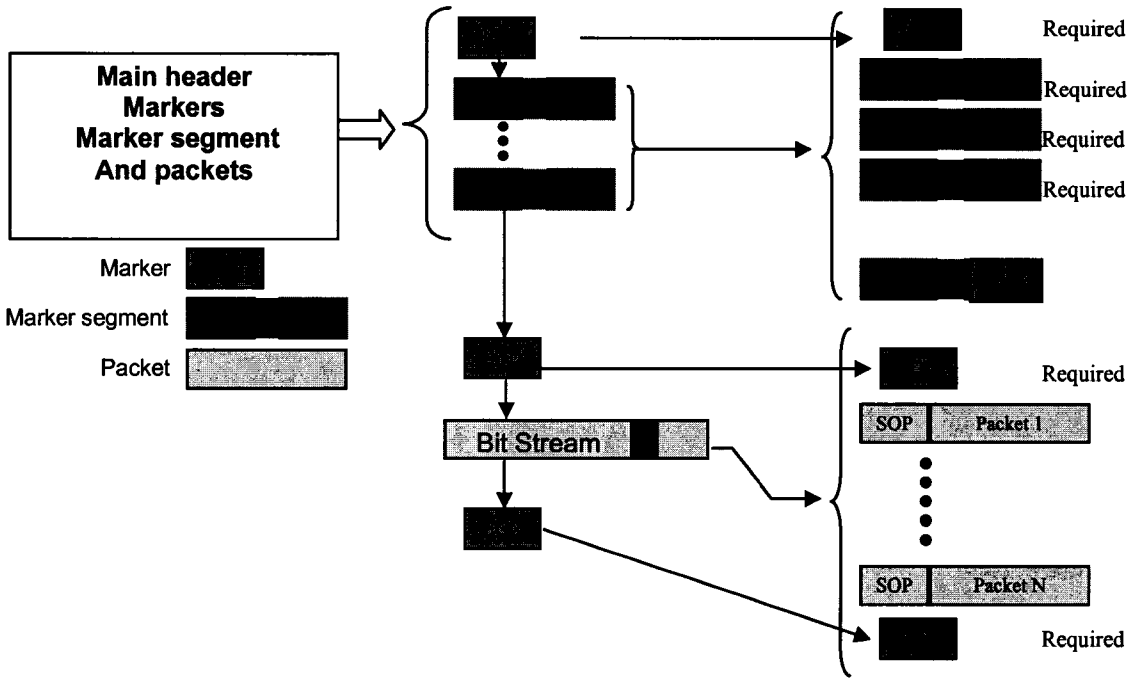



Figure 4.5: Final Code Streams

As previously explained in detail in Chapter 3, the final code stream for a single image contains the required and optional markers and segment markers as follows:

- The required delimiting markers

Each image code stream has only one Start of Code stream (SOC) marker, one Start of Data (SOD) marker and one End of Code stream (EOC) marker.

- The required fixed information marker segment

Image and tile size (SIZ)

- The required functional marker segments

Coding style default (COD) - Progression order

Quantization default (QCD)

- The optional pointer marker segments

Generally, pointer marker segments either provide a length or a pointer into the code stream.

Packed packet headers, main header (PPM): This marker presents a collection of the packet headers so that multiple reads are not required to decode headers. Also, extra protection can be applied to image headers including packet headers.

- In bit stream marker segments

The marker segments differ from all the others because they have no length field and they are found in the bit stream, not the main or a tile-part header.

Start of packet (SOP): The marker Marks the beginning of a partition and the index of that partition within a code stream.

End of packet header (EPH): The marker indicates the end of the packet header for a given packet. This delimits the packet headers in the PPM marker segments.

# Chapter 5

# Experimental Results

## 5.1. Introduction

The new ISO /ITU- T standard for still image coding, JPEG2000, has been shown to provide superior coding efficiency to the previous standard. In the first section, to judge the efficiency of the JPEG2000 as compared to the JPEG standard, extensive objective (PSNR, MSE) and subjective comparisons have been conducted with regard to lossy performance and in various compression ratios or bit-rates (bpp). The algorithms have been evaluated on several images, mainly from the JPEG2000 test set, which cover a wide range of imagery types.

Consequently, it is reasonable to argue that Motion-JPEG2000, the corresponding moving picture coding standard of JPEG2000, has an equally outstanding performance. In our performance studies of Motion-JPEG2000 in the second part, three aspects of Motion-JPEG2000 were examined: compression efficiency, error resilience, and image quality. These three aspects are chosen because they are crucial in visual communication.

In the third part, I break down MPEG-4 and observe that MPEG-4 has strong inter frame relation. Also in this section, I analyze the performance of Motion JPEG 2000 in error prone transmissions. I compare it to the motion JPEG video coding scheme. Motion JPEG2000 offers a number of very compelling advantages with respect to objective and subjective quality when compared to motion JPEG video coding. Eventually I conclude that MJPEG2000 is a suitable choice for wireless applications.

I conducted our experiments using JasPer, an implementation of the JPEG2000 standard from the University of British Columbia. The JasPer software has been included in the JPEG-2000 Part-5 standard (i.e. ISO/IEC 15444-5) as an official reference implementation of the JPEG-2000 Part-1 codec.

## 5.2. JPEG2000 performance study

I implemented a Matlab file to compare the compression efficiency and objective image quality of JPEG2000 with that of JPEG. Tables 5.1 to 5.3 are used to illustrate the objective image qualities whereas Figure 5.2 exhibits the subjective image quality in my performance study.

I chose 6 different bmp format images with a display resolution of 320 × 240 pixels and bit rate of 8bpp since these are common specifications among images. The images are each of size 257 KB. They cover a wide range of content; in particular, these frames exhibit very different characteristics and span a wide range of coding complexity. Figure 5.1 depicts the 6 test images exploit in our experience and Table 5.1, Table 5.2, and Table 5.3 illustrate the results of the compression schemes in various bit rates. Here, I examine three different bit rates (1.5, 0.5, and 0.2) for each test image and compare the Mean Square Error, Peak Signal-to-Noise Ratio and compression ratio in each table respectively.

The results from Table 5.1, Table 5.2, and Table 5.3 clearly show that the JPEG2000 performance outweighs JPEG objectively. Also, I observe a greater subjective quality of JPEG2000 compared to JPEG at higher compression rates in Figure 5.2.

Figure 5.1: Test images: Crowd, Lighthouse, Girl face, Bridge, Couple and Houses

| **Test Images (1.5 bpp)** | JPEG (MSE) | JPEG (PSNR) | JPEG (SIZE/Ratio) | JP2 (MSE) | JP2 (PSNR) | JP2 (SIZE/Ratio) |
|---|---|---|---|---|---|---|
| Crowd | 8.8 | 38.7db | 48.4KB/5.3 | 3.6 | 42.5db | 47.9KB/5.36 |
| Lighthouse | 24.4 | 34.3db | 47.7KB/5.38 | 11.1 | 37.7db | 47.8KB/5.37 |
| Girl face | 4.7 | 41.4db | 48.2KB/5.33 | 2.2 | 44.6db | 47.8KB/5.37 |
| Bridge | 58.1 | 30.5db | 48.2KB/5.33 | 29.7 | 33.4db | 47.8KB/5.37 |
| Couple | 14.5 | 36.5db | 48.1KB/5.34 | 7.3 | 39.5db | 47.9KB/5.36 |
| Houses | 49.1 | 31.2db | 47.9KB/5.36 | 20.3 | 35.1db | 47.9KB/5.36 |

Table 5.1: MSE and PSNR results for Test Images in bpp 1.5

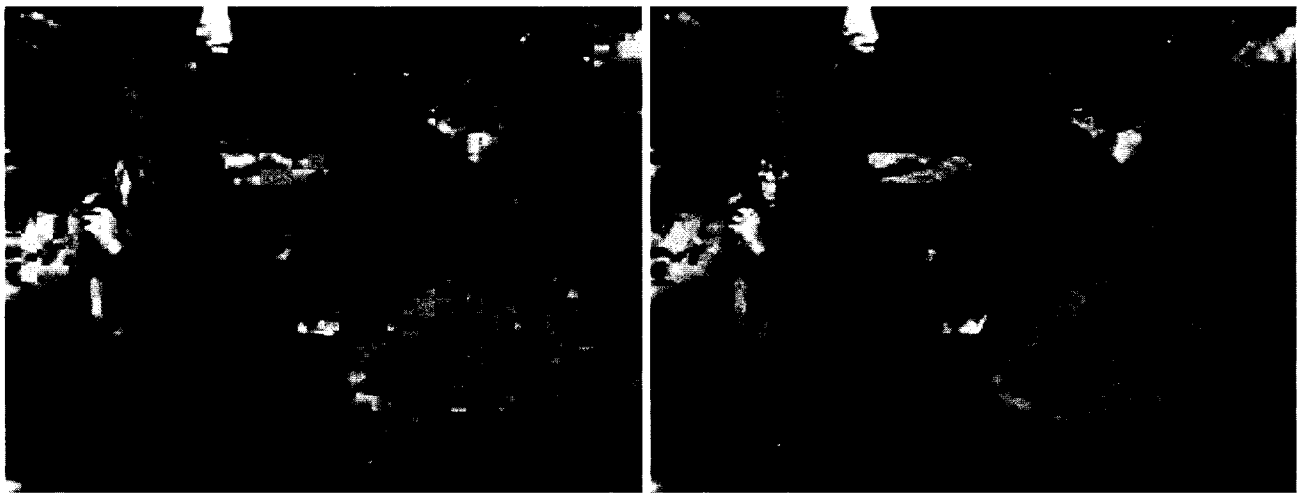
| **Test Images (0.5 bpp)** | JPEG (MSE) | JPEG (PSNR) | JPEG (SIZE/Ratio) | JP2 (MSE) | JP2 (PSNR) | JP2 (SIZE/Ratio) |
|---|---|---|---|---|---|---|
| Crowd | 46.2 | 31.5db | 15.9 KB/16.16 | 28 | 33.7db | 15.9 KB/16.16 |
| Lighthouse | 96.3 | 28.3db | 15.7KB/16.37 | 56.1 | 30.6db | 15.9 KB/16.16 |
| Girl face | 16.2 | 36db | 15.8KB/16.26 | 8.7 | 38.8db | 15.9 KB/16.16 |
| Bridge | 168.4 | 25.9db | 16.3KB/15.57 | 124.3 | 27.2db | 15.9 KB/16.16 |
| Couple | 59.2 | 30.4db | 15.9 KB/16.16 | 35.3 | 32.7db | 15.9 KB/16.16 |
| Houses | 226.2 | 24.6db | 15.7KB/16.37 | 139.4 | 26.7db | 15.9 KB/16.16 |

Table 5.2: MSE and PSNR results for Test Images in bpp 0.5

| **Test Images (0.2 bpp)** | JPEG (MSE) | JPEG (PSNR) | JPEG (SIZE/Ratio) | JP2 (MSE) | JP2 (PSNR) | JP2 (SIZE/Ratio) |
|---|---|---|---|---|---|---|
| Crowd | 219 | 24.7db | 6.01 KB/42.76 | 84.1 | 28.9db | 6.38 KB/40.28 |
| Lighthouse | 246.2 | 24.2db | 6.68KB/38.47 | 125 | 27.2db | 6.38 KB/40.28 |
| Girl face | 54.8 | 30.7db | 6.46KB/39.78 | 22.8 | 34.6db | 6.38 KB/40.28 |
| Bridge | 378.9 | 22.3db | 6.26KB/41.05 | 237.8 | 24.4db | 6.39 KB/40.21 |
| Couple | 185.6 | 25.4db | 6.41KB/40.09 | 92.7 | 28.5db | 6.34KB/40.53 |
| Houses | 590.2 | 20.4db | 6.9KB/37.24 | 350.9 | 22.7db | 6.39 KB/40.21 |

Table5.3: MSE and PSNR results for Test Images in bpp 0.2

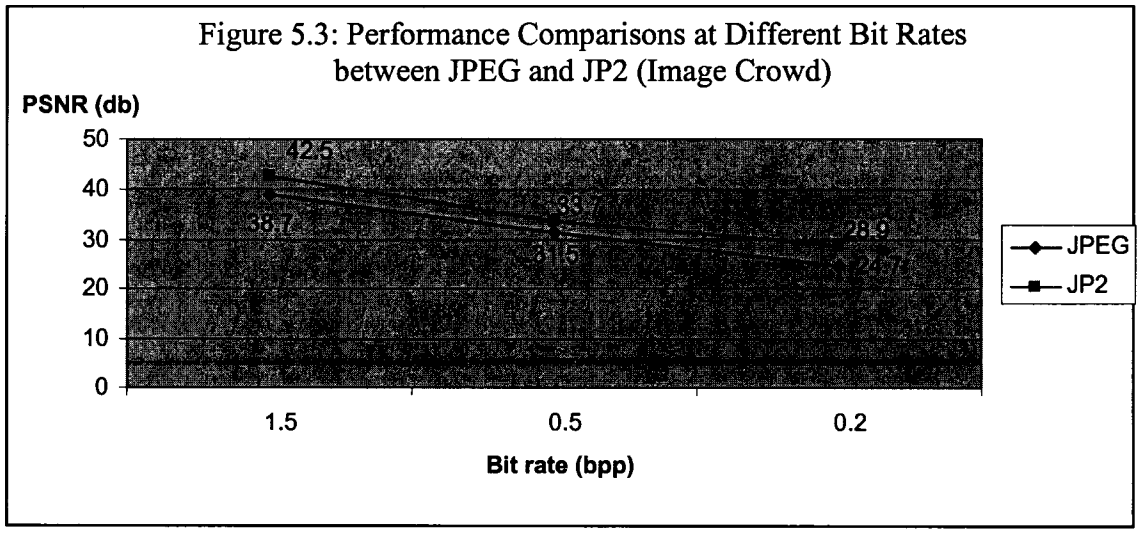
(a)



(b) (c)

Figure 5.2: Reconstructed Images Compressed at 0.2 bpp by means of (a) Original (b) JPEG and (c) JPEG2000

Figure 5.3 clearly implies that JPEG2000 outperforms JPEG in the same bit rate with respect to fidelity performance, and the PSNR improves even further in lower bit rates. In fact, JPEG2000 works better in low bit rates with a higher compression ratio. This is especially helpful in limited bandwidths such as wireless environments where the links are usually slow.



In the following part, I investigate the subjective and objective video streaming of JPEG2000, Motion JPEG2000. I chose a sequence of images ("Car.bmp") with a color depth of 8 and comprised of 20 frames. I then calculated each frame's PSNR as shown in Table 5.4 and Figure 5.4 respectively. As becomes apparent, Motion JPEG2000 gives better results in the low bit-rates (0.2 bpp) required in wireless environments which are bandwidth constrained.

| Frame Number | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PSNR(db)/bpp=0.5(JPEG) | 32.3 | 39.7 | 44.3 | 45.7 | 46.5 | 46.5 | 46.4 | 46.4 | 45 | 43.6 |
| Size(Bytes) | 6,306 | 6,358 | 6,392 | 6,326 | 6,362 | 6,344 | 6,349 | 6,388 | 6,372 | 6,375 |
| PSNR(db)/bpp=0.5(MJP2) | 34.4 | 35.4 | 36 | 36.5 | 36.9 | 37.2 | 37.2 | 37.5 | 37.4 | 37.5 |
| Size(Bytes) | 6,327 | 6,336 | 6,330 | 6,336 | 6,302 | 6,330 | 6,320 | 6,308 | 6,336 | 6,326 |
| Frame Number | | | | | | | | | | |
| PSNR(db)/bpp=0.2(JPEG) | 24.8 | 25.4 | 25.5 | 25.7 | 25.8 | 25.8 | 25.8 | 25.8 | 25.7 | 25.7 |
| PSNR(db)/bpp=0.2(MJP2) | 28.7 | 29.7 | 29.9 | 30.3 | 30.5 | 30.9 | 30.8 | 30.9 | 30.8 | 31 |
| Frame Number | | | | | | | | | | |
| PSNR(db)/bpp=0.5(JPEG) | 6,308 | 6,346 | 6,293 | 6,361 | 6,348 | 6,336 | 6,393 | 6,370 | 6,328 | 6,378 |
| Size(Bytes) | 38.6 | 36.8 | 37.1 | 37.6 | 38.8 | 40.8 | 42.5 | 43.6 | 42.9 | 42.6 |
| PSNR(db)/bpp=0.5(MJP2) | 6,291 | 6,331 | 6,257 | 6,303 | 6,299 | 6,320 | 6,331 | 6,321 | 6,332 | 6,317 |
| Size(Bytes) | 37.4 | 37.4 | 37.8 | 37.7 | 37.8 | 37.8 | 38 | 37.8 | 37.9 | 37.8 |
| Frame Number | | | | | | | | | | |
| PSNR(db)/bpp=0.2(JPEG) | 25.6 | 25.7 | 25.7 | 27.1 | 25.8 | 25.8 | 25.9 | 25.9 | 25.8 | 25.8 |
| PSNR(db)/bpp=0.2(MJP2) | 30.9 | 31.1 | 31.4 | 31.2 | 31.2 | 31.1 | 31.1 | 31.2 | 31.5 | 31.2 |

Table 5.4: Comparison of JPEG and JPEG2000 Video Streaming Quality

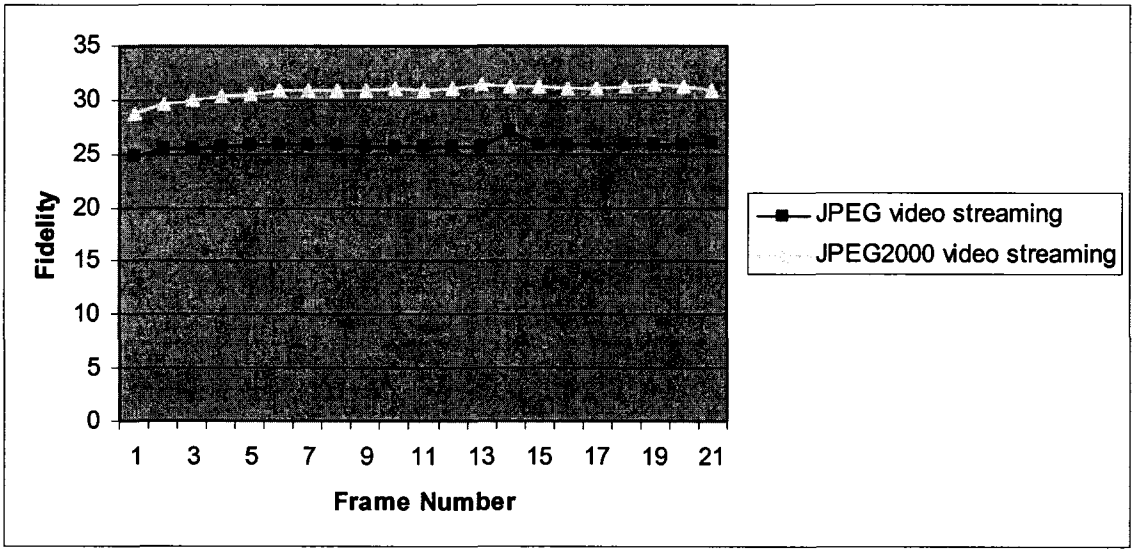



Figure 5.4: Chart Depicts Superior Quality of JPEG2000 over JPEG Video Streaming

Also in Table 5.5, we can clearly observe that the frame order in the received code stream does not match the display order. This results in greater delay and interdependency between frames, neither of which is desirable in a real time streaming and error prone wireless environment.

| Bit Stream Frame order No | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Frame Type | I | P | B | P | B | P | B | P | B | P |
| Display Frame order No | 1 | 3 | 2 | 5 | 4 | 7 | 6 | 9 | 8 | 11 |
| SNR(Y) db | 37.9 | 36.5 | 36.4 | 36.8 | 36.3 | 37 | 36.4 | 37.2 | 36.5 | 37.5 |
| Bit-rate | 67984 | 29064 | 7808 | 29280 | 7824 | 31696 | 7664 | 32904 | 8216 | 36176 |
| | | | | | | | | | | |
| Bit Stream Frame order No | | | | | | | | | | |
| Frame Type | B | P | B | P | B | P | B | P | B | P |
| Display Frame order No | 10 | 13 | 12 | 15 | 14 | 17 | 16 | 19 | 18 | 21 |
| SNR(Y) db | 36.7 | 38 | 36.9 | 37.5 | 36.7 | 37.5 | 36.8 | 37.4 | 36.7 | 37.3 |
| Bit-rate | 8872 | 31408 | 10720 | 33280 | 10200 | 33504 | 10336 | 35360 | 10592 | 34592 |

Table 5.5: Inter frame Dependency between Consequent Frames

Finally, to demonstrate the error resilience feature of JPEG2000 I examined the sequence of frames ("driving.avi") with $BER = 10^{-3}$ caused by error prone communication channels and compared the performance of the JPEG2000 with the JPEG scheme. We can clearly observe the greater quality objectively (Figure 5.5) and subjectively (Figure 5.6 a, b). I noticed that in my experience, the JPEG image decoder crashed often with erroneous images and could not display any image at all.

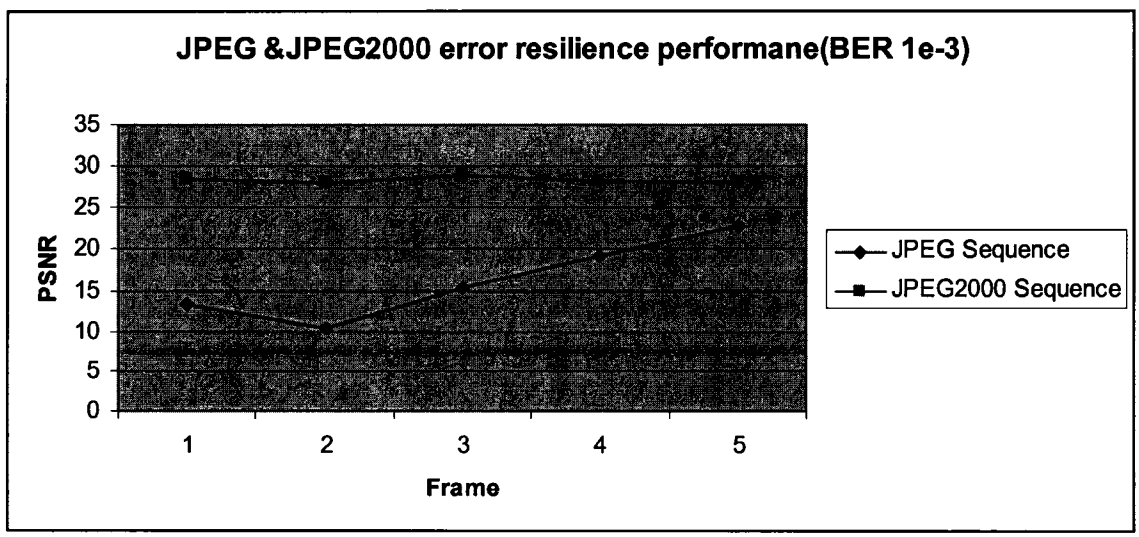
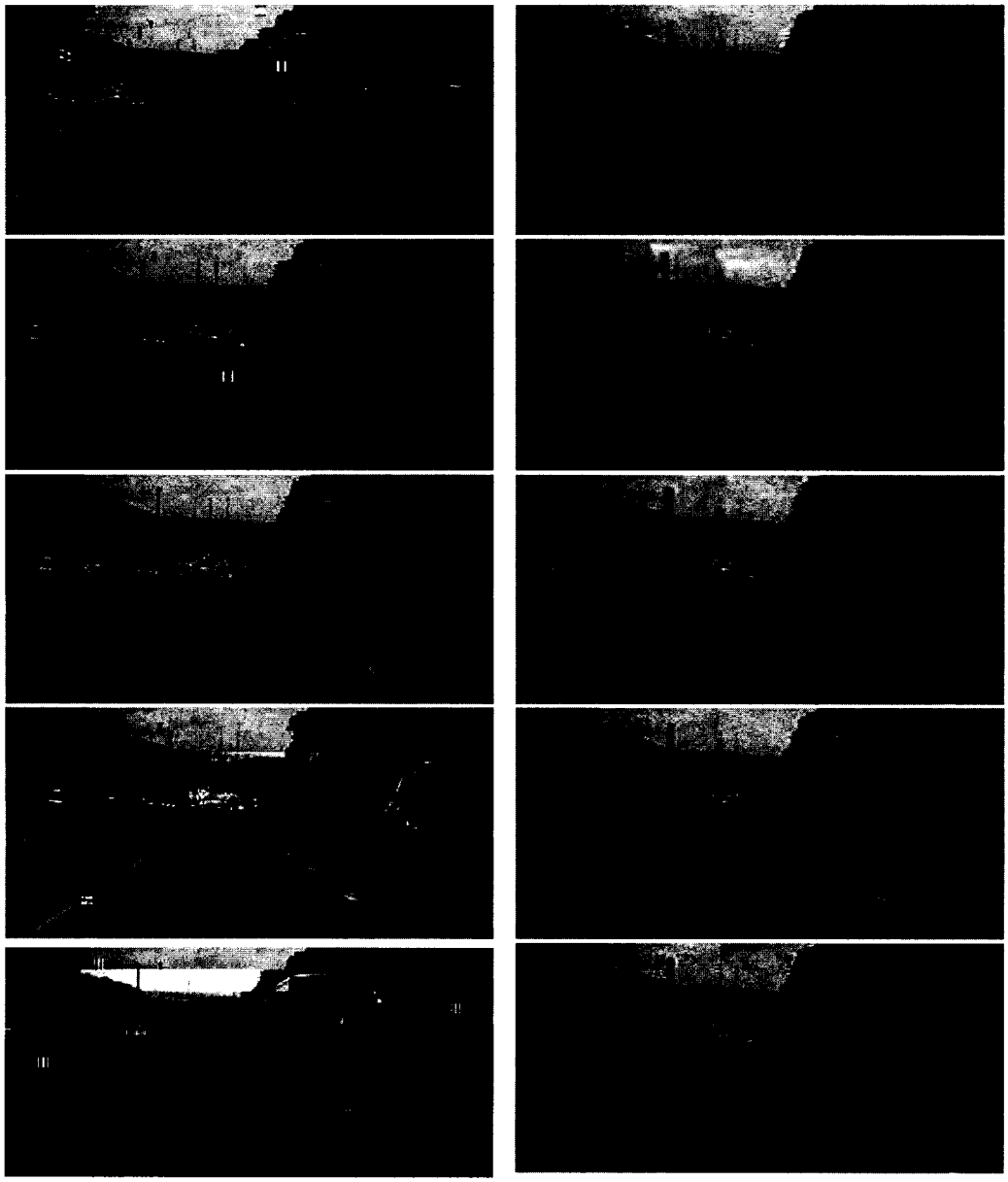



Figure 5.5: Error Resilience Performance of Video Streaming



(a) (b)

Figure 5.6: Sequences of Frames Illustrates Quality Superiority of (b) JPEG2000 over (a) JPEG Video Streaming in $BER = 10^{-3}$

# Chapter 6

# Conclusions and future work

In order to improve wireless video streaming there are several areas of research that can be investigated further. One of the most prominent issues among the wireless video streaming areas of research is robustness to error or error resilience.

In fact, error resilience in wireless video streaming is becoming an area of extensive research due to the error-prone nature of radio channels with variations in throughput and delay. We believe that real time wireless multimedia lacks the unifying framework of a source and channel coding integrated approach to incorporate error resilience into packet video streaming over the Internet. This thesis effectively addresses the topic of the robustness to error of data streams over wireless links.

## 6.1. Conclusions of the Thesis

The purpose of this thesis was to propose an integrated error resilient scheme that is backward compatible with JPEG2000 image streams, and can be summarized as follows:

- I modified and implemented the light and portable version of the Jasper (reference implementation of the codec specified in the JPEG-2000 Part-1 standard) for the purpose of wireless streaming, and also employed the packetization of the JPEG2000 code stream with SOP signals inside the header.
- Small packet size and greater resolution are deployed by signalling the COD marker for smaller precinct size. In fact, increasing the number of resolutions results in spacing the cumulative layer sizes more closely, and each layer and each precinct packet represent a smaller incremental contribution to the image quality. This is specifically useful in the SEGMARK option to improve image quality in the presence of channel errors in the code stream

- I utilized RTP / UDP-Lite / IP / MAC as the underlying protocol to deliver real time video streams from the server to the client over a wireless network, and to define a payload format for JPEG2000 video streams over the Real-time Transport Protocol. I then presented the format to incorporate JPEG2000 Error resilient schemes at the source and packet levels into the RTP-based transmission of JPEG2000 wireless imaging.
- The JPEG2000 code stream is wisely inserted inside the RTP packet with the upper range of 1500 octets (MTU), resulting in the minimum amount of fragmented JPEG2000 in each RTP packet.
- An unequal error protection scheme based on Reed-Solomon FEC is offered by the JPEG2000 RTP payload header to detect and correct erroneous data on the fly for each JPEG2000 packet, whereas by default the header and finest quality layer have a higher amount of redundancies and protection
- An algorithm is proposed for reordering received RTP packets and RTP packet loss validation
- I proposed the component progressive (Component-Resolution-Precinct-Quality) data format in which, upon the detection of each corrupted quality packet, the packet and consequence related quality component will be discarded. This approach improves the quality of the image and prevents further distortion of the image, also decreasing data overhead.
- In the experimental results section, the superiority of my proposal over different standards is objectively (PSNR) and subjectively (Image) judged in various bpp or compression ratios by graphs and figures.
- I also compared the achieved fidelity of Motion JPEG2000 with regard to MPEG4/H264 and concluded the benefits of higher error resilience in my proposal.
- We can observe that the Motion JPEG2000 image stream occupies more bandwidth than the MPEG4, which is a fair price to pay for higher error resilience in the code stream when operating on channels with very high noise levels.

## 6.2. Future work

As previously indicated, JPEG2000 has several features that offer scalability, a number of qualities of services, and revenue strategies. These features can initiate many research activities regarding wireless multimedia. Features like ROI can be utilized to construct a code stream based on the semantic content of the video. The error resilience can be incorporated into the semantic content of the data and thus produce a reliable and optimized video stream.

In fact, with the growing omnipresence and mobility of multimedia-enabled devices, emerging metadata-driven multimedia access applications are playing a significant role in the next generation and future of multimedia communication applications. By the inclusion of metadata in JPEG2000 encoded images we can indicate to a server or gateway what version of the image to deliver to a limited client. JPEG2000 specifies a region-of-interest description box in its file format, and provides a wide range of other metadata.

Yet another interesting area of research is the integration of video watermarking in Motion JPEG2000 to protect the originality and authentication of multimedia information and robustness. In spite of the number of research efforts that have been invested in this field, at this point there is still no solution for robust watermarking.

# Bibliography


[1] M. D. Adams. "The JPEG-2000 still image compression standard." *ISO/IEC JTC 1/SC 29/WG 1 N 2412*, September 2001.http://www.ece.uvic.ca/~mdadams and distributed with the JasPer software.

[2] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1737–1744, Nov. 1996.

[3] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control, " *Internet Engineering Task Force*, RFC 2581, April 1999.

[4] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans.* Image Processing, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[5] B.A. Banister, B. Belzer, T.R. Fischer, "Robust image transmission using JPEG2000 and turbo-codes," *IEEE Signal Processing Letters*, vol. 9, n. 4, pp. 117-119, Apr. 2002.

[6] C. Berrou, A. Glavieux, "Near-optimum error correcting coding and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, n. 10, pp. 1261-1271, Oct. 1996.

[7] C. Berrou, A. Glavieux, P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," *in Proc. IEEE lnt. Conf on Communications*, pp. 1064-1070, Geneva, Switzerland, May 1993

[8] M. Boliek, C. Christopoulos and E. Majani, "JPEG2000 Part I Final Draft International Standard," *ISO/IEC FDIS15444-1, ISO/IEC JTC1/SC29/WG1 N1855*, August 18, 2000.

[9] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inform. Control*, vol. 3, pp. 68-79, March 1960.

[10] A. Boukerche, T. Huang, R. W. N. Pazzi, "A real-time transport protocol for image-based rendering over heterogeneous wireless networks," *MSWiM 2005*: 333-340

[11] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.L. Yeo, "Wavelet transforms that map integers to integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, July 1998.

[12] Call for contributions for JPEG 2000 (JTG 1.29.14, 15444): *Image coding system. Technical Report N505*, ISO/IEG JTGl/SG29/WGl, March 1997

[13] H. Dong, D. Chakares, A. Gersho, E. Belding-Royer, and J. Gibson, "Selective bit-error checking at the MAC layer for voice over mobile ad hoc networks with IEEE 802.11,"*Proc. IEEE Wireless Conference (WCNC)*, Atlanta, USA, , pp. 1240–1245. 2004

[14] F. Dufaux and T. Ebrahimi, "Motion JPEG2000 for wireless applications," *in Proceedings of 1st International JPEG2000 Workshop*, pp. 2036–2039, Lugano, Switzerland, July 2003.

[15] E. Edwards, S. Futemma, E. Itakura, N. Tomita, A. Leung, T. Fukuhara, "RTP payload format for JPEG 2000 video streams," *WG1N2768.*

[16] F. Frescura, M. Giorni, C. Feci and S. Cacopardi, "JPEG 2000 and MJPEG 2000 transmission in 802.11 wireless local area networks", *IEEE Trans. on Consumer Electronics*, vol. 49, no. 4, pp. 861-871, Nov. 2003.

[17] A. J. Goldsmith, M. Effros "Joint design of fixed-rate source codes and multiresolution channel codes," *IEEE Trans. Comm.*, 46(10):1301–1312, 1998.

[18] M. Grangetto, E. Magli and G. Olmo, "Error sensitivity data structures and retransmission strategies for robust JPEG 2000 wireless imaging", *IEEE Trans. on Consumer Electronics*, vol. 49, no. 4, pp. 872-882, Nov. 2003.

[19] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, n. 4, pp. 389-400, Apr. 1988.

[20] J. Hagenauer, N. Seshadri, C.-E.W. Sundberg, "The performance of rate-compatible punctured convolutional codes for digital mobile radio," *IEEE Transactions on Communications*, vol. 38, n. 7, pp. 966-980, Jul. 1990

[21] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. on Communications*, 36, 4:389– 400, 1988.

[22] F. Hekland," A Review of Joint Source-Channel Coding" *Norwegian University of Science and Technology (NTNU)*, February 16, 2004

[23] IEEE 802.11 WG, "Draft supplement to standard for telecommunications and information exchange between systems - LAN/MAN specific requirements - part 11: Wireless medium access control (MAC) enhancements for quality of service (QoS)," *IEEE 802.11e/D5.0*, August 2003.

[24] ISO/IEC 15444-3:2002, "Information technology – JPEG 2000 image coding system – Part 3: Motion JPEG 2000," 2002.

[25] ISO/IEC 14496-1:2004, "Information technology – Coding of audio-visual objects – Part 1: Systems," 2004.

[26] ISO/IEC 15444-1:2000, "Information technology—JPEG 2000 image coding system—Part1: Core coding system," 2000.

[27] ISO/IEC 14492-1:2000, "Lossy/lossless coding of bi-level images," 2000.

[28] ISO/IEC JTC1/SC29/WG1 WG1N3294, "JPEG 2000 image coding system – Part 11: Wireless JPEG 2000" *Working Draft version 3.1*, April 2004.

[29] ISO/IEG JTGl/SG29/WGl, New work item proposal, "JPEG2000 image coding system," *Technical Report N390,* June 1996.

[30] V. Jacobson. "Congestion Avoidance and Control," *Proceedings of ACM SIGCOMM '88*, Stanford, CA, August 1988.

[31] A. Jacquin. "Fractal image coding based on a theory of iterated contractive image transformations," *SPIE Visual Communications and Image Processing*, volume 1360, pages 227–239, 1990.

[32] S. Khayam, S. Karande, H. Radha, and D. Loguinov, "Performance analysis and modeling of errors and losses over 802.11b LANs for high-bitrate real-time multimedia," *Signal Processing: Image Communication*, vol. 18, no. 7, pp. 575–595, August 2003.

[33] L.A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst, "The UDP-lite protocol," *draft-ietf-tsvwg-udp-lite-02.txt*, August 2003.

[34] C. Leicher, "Hierarchical encoding of MPEG sequences using priority encoding transmission (PET)," *ICSI, Tech. Rep. TR-94-058*, Nov. 1994.

[35] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 244–250, Apr. 1992.

[36] S. Lin, D. J. Costello, and M. J. Miller, "Automatic repeat-request error control schemes," *IEEE Communications Magazine*, 12:5–17, December 1984.

[37] J. W. Modestino, D. G. Daut, "Combined source-channel coding of images," *IEEE Trans. Comm.*, COM-27:1644–1659, 1979.

[38] A.E. Mohr, R.E. Ladner, E.A. Riskin, "Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction," *IEEE Journal on Selected Areas in Communications*, vol. 18, n. 6, pp. 819-828, Jun. 2000.

[39] A. Natu and D. Taubmann, "Unequal protection of JPEG2000 code-streams in wireless channels", *In Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, Nov. 2002, vol. 1, pp. 534.538.

[40] D. Nicholson, C. Lamy-Begot, X. Naturel and C. Poulliat, "JPEG 2000 backward compatible error protection with Reed-Solomon codes", *IEEE Trans. on Consumer Electronics*, vol. 49, no. 4, pp. 855-860, Nov. 2003.

[41] J. Postel, "User Datagram Protocol," *Request for Comments RFC 768*, ISI, August 1980.

[42] Poulliat C., Vila P., Pirez D., Fijalkow I., "Progressive JPEG2000 Image Transmission over noisy channel", *Eusipco 2002*, 3rd-6th September 2002.

[43] M. J. Ruf, J. W. Modestino, "Operational rate-distortion performance for joint source and channel coding of images," *IEEE Trans. Image Proc.*, 8(3):305–320, 1999.

[44] V. Sanchez and M.K. Mandal, "Robust transmission of JPEG 2000 images over noisy channels," *Proceedings of IEEE ICCE'02*, pp. 80- 81, 2002.

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

[46] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real Time Applications," *RFC 3550*, Internet Engineering Task Force, July 2003.

[47] A. Servetti and et al., "Error Tolerant MAC Extension for Speech Communications over 802.11 WLANs," *IEEE VTC*, 2005.

[48] A. Servetti and J.C. De Martin, "Link-level unequal error detection for speech transmission over 802.11 networks," *in Proc. Special Workshop in Maui - Lectures by Masters in Speech Processing*, Maui, Hawaii, USA, January 2004.

[49] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, July, October, 1948.

[50] L. Shu, and D. Costello, Jr. , "Error Control Coding: Fundamentals and Applications," *Englewood Cliffs, N.1.: Prentice-Hall*, 1978

[51] A. Skodras, C. Christopoulos and T. Ebrahimi "The JPEG 2000 still image compression standard", *IEEE Signal Processing Magazine* , vol. 18, no. 5, pp. 36 - 58, Sept. 2001.

[52] D. Taubman, "High performance scalable image compression with EBCOT," *in Proc. of IEEE International Conference on Image Processing*, vol. 3, pp. 344–348. , 1999

[53] D. S. Taubman, "Remote browsing of JPEG2000 images," *Proc. of 2002 IEEE Inter. Conf. Image Processing*, vol. 1, pp. 229 , Sept. 2002

[54] A. J. Viterbi, J. K. Omura, "Principles of Digital Communications and Coding," *McGraw-Hill*, New York 1979.

[55] L. Weiliang, D.G. Daut," An adaptive UEP transmission system for JPEG2000 codestream using RCPT codes" *Signals, Systems and Computers*, vol 2, 7-10 Nov. 2004

[56] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

# Appendix A

# Light JPEG2000 compression software

```
/*
Babak Simaie
PARADISE Lab
University of Ottawa,
July 4, 2005

This program converts ppm to error resilient jpc

*/

#include "jasper.h"


int main(int argc, char *argv [])
{
      jas_image_t *image;
      jas_stream_t *in;
      jas_stream_t *out;
      jas_stream_t *packet_stream;
      jas_stream_t *packet[];
      int i;

      if(argc<2){
            printf("Please specify a file!\n");
            return 0;
      }

      /*open input file with read access type */
      if(!(in = jas_stream_fopen(argv[1],"rb"))){
           fprintf(stderr, "error: cannot open output image file %s\n",
argv[1]);
            exit(EXIT_FAILURE);
      }

      /* open output file with write access type*/
      if(!(out = jas_stream_fopen("test.jpc", "w+b"))){
           fprintf(stderr, "error: cannot open output image file %s\n",
"test");
            exit(EXIT_FAILURE);
      }

      /*create an image from coded data in a stream*/
      if(!(image = pnm_decode(in, ""))){
            fprintf(stderr, "error: cannot load image data\n");
            exit(EXIT_FAILURE);
      }

      /*emit the coded version of an image to a stream*/
      if(jpc_encode(image,out,"sop\neph")){
            fprintf(stderr, "error: cannot encode image\n");
```

```
            exit(EXIT_FAILURE);
      }

      if(!(in=jas_stream_fopen("test.jpc","rb"))){
            fprintf(stderr, "error: cannot open image file %s\n",
"test.jpc");
            exit(EXIT_FAILURE);
      }

      for(i=0;;i++){
            if(!(packet_stream=jpc_fragment(in))){
                  fprint(stderr, "error: cannot fragment the
packets\n");
            }     exit(EXIT_FAILURE);
            packet[i]=packet_stream;
      }


      jas_image_destroy(image);

      return EXIT_SUCCESS;

}
```