The Design, Performance Analysis and Application of an MPEG-2 Transcoder

A thesis

Submitted in Partial Fulfillment of the requirements for the Degree Master of Science in the Department of Computer Science at the Ohio state University

By

Vinod Krishnan Kulathumani

The Ohio State University

2001

Approved By:

Master's examination Committee:

Prof. Wu-Chi Feng, Advisor

Prof. Arian Durresi

Advisor

Computer Science Program

ABSTRACT

This thesis describes the design, implementation and performance analysis of a transcoder for MPEG– 2 streams. When streaming MPEG video over a heterogeneous network such as Internet, the available bandwidth is not the same at different points of the broadcast chain. Video transcoding is one of the key technologies required to dynamically adapt the bit-rate of the video stream to the available bandwidth along a heterogeneous network. However, in the process of conversion from higher bit rate to a lower bit rate, some delay is introduced in the bit stream. In interactive applications such as video conferencing this delay has to be minimized. In this thesis, two methods for the transcoding of MPEG streams have been implemented. The first is by increasing the quantization scale and the other by dropping of B-frames. Both the methods perform the task with minimum storage and computational requirements and hence introduce very less delay. The quantization scale is changed in the DCT domain itself. This means that the MPEG stream need not be decoded and then re-encoded completely. Also, the motion vectors are re-used and re-computation of motion vectors is avoided in the process. A potential application for such an MPEG transcoder as a bandwidth manager has been simulated. The bandwidth manager maintains the bit rate of the stream at the required level by using the transcoder. The transcoder responds to feedback from the manager and changes the size of the stream and maintains the required bit rate. The transcoder can be

used in application gateways, especially when the devices at the other end are lap tops and palm tops, which do not have the processing capacity to handle the full stream quality or if the connection is a low bandwidth dial-up connection.

DEDICATION

Dedicated to My Parents, who have directed me to this path

ACKNOWLEDGMENTS

I remain grateful to Prof. Wu Chi Feng for having given me an opportunity to work under him and for guiding me through this work. I am also thankful to Prof. Arian Durresi for being on my Master's examination committee.

Above all, I would like to thank my parents, for creating an environment in which following this path seemed so simple and natural.

VITA

November 7, 1977 ……..	**Born – Tamil Nadu, India**
1999 ……..	**B.E.., Computer Science** **VJTI, Mumbai, India**

PUBLICATIONS

1. N. Chandhok, A. Durresi, R. Jagannathan, R. Jain, S. Seetharaman, K. Vinodkrishnan, "IP over Optical Networks: A Summary of Issues", Internet Draft draft-osu-ipo-mpls-issues-00.txt, Work in progress, July 2000.
2. K.Vinodkrishnan, “Fault Restoration in IP over Optical networks”, Accepted by *Journal on high Speed Networks*, Jan 2001.
3. K.Vinodkrishnan, “VOIP: Products, Services and Issues", Dec 1999, http://www.cis.ohio-state.edu/~jain/cis788-99/voip_products/index.html

TABLE OF CONTENTS

ABSTRACT ..... **II**

DEDICATION ..... **IV**

ACKNOWLEDGMENTS ..... **V**

VITA ..... **VI**

PUBLICATIONS ..... **VI**

TABLE OF FIGURES ..... **X**

INTRODUCTION **1**

1.1 BACKGROUND 1

1.2 RESEARCH OBJECTIVES 2

1.3 OUTLINE OF THE THESIS 5

BACKGROUND INFORMATION **6**

2.1 MPEG-2 6

2.1.1 The Standard *6*

2.1.2 Picture Types *7*

2.1.3 The Architecture *7*

2.2 THE VIDEO TRANSCODER 10

2.2.1 Basic architecure *10*

2.2.2 Reducing complexity of the model *13*

2.3 MODES OF TRANSCODING 13

2.3.1 Rate reduction *13*

2.3.2 Temporal Transcoding *14*

2.3.3 Resolution reduction *14*

2.4 CONCERNS WITH MPEG-2 STREAMING 14

2.5 RELATED WORK 15

2.5.1 An application level Video gateway *15*

2.5.2 Video Transcoding for Universal Multimedia access *16*

2.5.3 An MPEG-1 Transcoder *16*

2.5.4 Reducing drift errors *16*

DESIGN AND IMPLEMENTATION OF THE TRANSCODER 18

3.1 INCREASING QUANTIZATION SCALE 18

3.2 BLOCK DIAGRAM 19

3.3 DESIGN ISSUES 22

3.4 B-FRAME DROPPING 22

PERFORMANCE ANALYSIS OF THE MPEG-2 TRANSCODER 23

4.1 REDUCTION IN FRAME SIZES 23

4.1.1 Comparison of bit rates *28*

4.2 FRAME PROCESSING TIMES 29

4.3 QUALITY 32

4.4 SUMMARY OF FINDINGS 34

THE ADAPTIVE BANDWIDTH MANAGER : AN APPLICATION 35

5.1 THE BANDWIDTH MANAGER 35

5.1.1 Proxy side placement 35

5.1.2 Client side Management 38

5.2 BANDWIDTH MANAGER PERFORMANCE ANALYSIS 39

5.2.1 Manager at the Gateway 39

5.2.2 Bandwidth manager at the client 46

5.3 DIFFERENCES BETWEEN HAVING THE MANAGER AT THE CLIENT AND AT THE GATEWAY 48

CONCLUSION 49

6.1 CONTRIBUTIONS 49

6.2 FUTURE WORK 50

REFERENCES 51

TABLE OF FIGURES

Figure 2.1 : MPEG-2 Encode – Decode architecture……. 8

Figure 2.2 : MPEG-2 Stream structure…….. 9

Figure 2.3 : Basic transcoder architecture…….. 11

Figure 2.4 : Block diagram of a detailed transcoder…….. 12

Figure 3.1 : Block diagram of our transcoder……. 19

Figure 3.2 : Processing of macroblock information…… 21

Figure 4.1 : Frame sizes with different transcoding rates for NBA.mpg…… 24

Figure 4.2 : Frame sizes with different transcoding rates for tennis.mpg……... 24

Figure 4.3 : I-frame sizes for NBA.mpg……. 25

Figure 4.4 : P-frame sizes for NBA.mpg…… 25

Figure 4.5 : B-frame sizes for NBA.mpg…… 26

Figure 4.6 : I-frame sizes for tennis.mpg…… 26

Figure 4.7 : P-frame sizes for tennis.mpg……... 27

Figure 4.8 : B-frame sizes for tennis.mpg…….. 27

Figure 4.9 : Comparison of bit rates in KBps for tennis.mpg……. 28

Figure 4.10 : Comparison of bit rates for NBA.mpg……. 29

Figure 4.11 : Frame processing time for NBA.mpg……... 29

Figure 4.12 : Frame processing times for tennis.mpg……. 30

Figure 4.13 : Average frames per second generated by transcoder…… 31

Figure 4.14 : Comparison of times taken to transcode and completely re-encode … 31

Figure 4.15 : An I-frame from tennis.mpg, no transcoding…… 32

Figure 4.16 : An I frame from tennis.mpg, transcode rate = 4…… 32

Figure 4.17 : An I frame from tennis.mpg, transcode rate = 8…… 32

Figure 4.18 : An I frame from tennis.mpg, transcode rate = 16……. 32

Figure 4.19 : A B frame from tennis.mpg, no transcoding……. 33

Figure 4.20 : A B frame from tennis.mpg, transcoding rate = 4…… 33

Figure 4.21 : A B frame from tennis.mpg, transcoding rate = 8……. 33

Figure 4.22 : A B frame from tennis.mpg, transcoding rate = 16……... 33

Figure 5.1 : Architecture for Proxy side bandwidth manager……. 36

Figure 5.2 : Block diagram for client side bandwidth manager…….. 39

Figure 5.3 : Buffer size required, without bandwidth manager…….. 40

Figure 5.4 : Buffer variations with transcoder used…… 41

Figure 5.5 : Frame rate and quality variation with transcoder employed……... 41

Figure 5.6 : Frame rate variations with only frame dropping……. 42

Figure 5.7 : Buffer required with only frame dropping…….. 43

Figure 5.8: Pseudo code for buffer management with transcoding and dropping….. 43

Figure 5.9 : Buffer variations with transcoder and dropper…… 44

Figure 5.10 : Frame rate and quality variations with transcoder and frame dropper.. 44

Figure 5.11 : Bandwidth trace showing bandwidth between 20 and 80 KBps…… 45

Figure 5.12 : Buffer size required, high bandwidth variation link……. 45

Figure 5.13 : Frame rate and quality variations, highly variable bandwidth link….. 46

Figure 5.14 : Number of frames received by client, without a bandwidth manager... 47

Figure 5.15 : Frame rate and quality variations with manager at client side…….. 47

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, various multimedia services, such as teleconferencing, video on demand, and distance learning have been emerging. However, a unified network infrastructure does not exist yet for those services and many different existing networks such as ATM, TCP/IP, Wireless and PSTN, etc. are interconnected resulting in a heterogeneous network environment for the delivery of those services. Therefore, at the gateways or proxies, a dynamic bit-rate adaptation of incoming video bit stream is needed to match the available bandwidth of outgoing network. Dynamic bit rate adaptation can be achieved using the scalable coding schemes provided in current video coding standards. However, it can only provide up to three levels of discrete video quality because of the limit on the number of enhancement layers. In many networked multimedia applications, a much finer scaling capability is desirable.

Moreover, devices such as laptops and palm devices have started using multimedia applications. These devices cannot get enough bandwidth or CPU power to

handle a very high quality stream or a large frames per second. Reducing the bit rate dynamically for these streams is highly desirable.

1.2 Research Objectives

Converting a previously compressed video bit-stream to a lower bit rate bit-stream through transcoding can provide finer and more dynamic adjustments of the bit rate of the coded video bit-stream. The objective of this thesis is to provide a fast real time mechanism to dynamically adapt a video stream to a heterogeneous bandwidth environment.

The process of converting between different compression formats and/or further reducing the bit rate of a previously compressed signal is known as transcoding, and can introduce significant impairments if performed without due care. Transcoding may be required at several points in the broadcast chain e.g. converting from an acquisition to an editing format or converting from a high bit rate studio format to a low bit rate distribution format. Transcoding differs from first generation coding in that a transcoder only has access to a previously compressed signal which already contains quantization noise compared to the original source signal.

However, in the process of conversion from higher bit rate to a lower bit rate, we may introduce some delay in the bit stream. In interactive applications such as video conferencing we would like to minimize this delay. Some common approaches for transcoding are:

a. Increasing the Quantization scale.

b. Dropping intermediate frames.

c. Re encoding by change of coding mode decisions.

d. Changing the Resolution.

However, changing the coding mode decisions and re-encoding the entire stream are time consuming and computationally intensive processes involving re-construction of motion vectors and coded blocks. Dropping of resolution involves going back into the pixel domain from the frequency domain. This involves a computationally expensive Inverse DCT computation for each frame. However, increasing the quantization scale and dropping of intermediate frames can both be performed in the DCT domain itself. The motion vectors and the coding mode decisions carried in the input stream can be re-used in both these methods.

This thesis gives the implementation details, performance analysis and applications of the two methods stated above. To vary the quantization scale, it is required to inverse quantize the individual frames and then re-quantize them. This requires some computation. Hence the performance will depend upon the processing environment. But it is to be noted that this processing is only $1/5^{th}$ of that required to encode an MPEG stream. Solaris and UNIX environments without MMX hardware transcode streams at about 15 frames per second. But systems with MMX hardware, which can encode MPEG streams at around 30 fps, can also transcode streams faster.

When re-quantizing the streams, the original motion vectors are re-used. However, the predictions become invalid now, as the regenerated picture is not the same as the one during the encoding process. This causes a “drift error” in the transcoder,

which is reset when an I frame comes along. Hence the quality of the generated images are not the same as they would be if the whole stream was re-encoded. The slight loss of quality is compromised for improving the performance.

Sometimes, it may be preferable to drop some frames from the stream instead of compromising the quality too much. Hence, a B-frame dropper has also been designed. Depending upon the requirements, only frames can be transcoded or only dropped or both. Only transcoding will generate a constant frames per second at the receiver but varying quality of frames. Only dropping of frames will maintain the quality of the frames but the frames per second will vary.

The transcoding times of the transcoder are compared with those for the complete re-encoding of the stream. The results promise use of the transcoder in real time scenarios. The reduction in frame sizes for various transcoding rates are examined. Quality of the transcoded stream is also compared with the transcoding rate.

A potential application for the transcoder as a bandwidth manager has been simulated. This manager can be located at the client side or along with the transcoder at a gateway. At the client side, it tries to maintain a constant incoming frame rate by sending feedback to the transcoder. When placed along with the transcoder, it maintains an appropriate buffer and prevents loss of data. When the bandwidth on the output side reduces, the buffer will tend to overflow. The bandwidth manager prevents this by using the transcoder and the B-frame dropper. It tries to balance the buffer at an optimum range without an overflow or underflow.

1.3 Outline of the thesis

The rest of the thesis is organized as follows. Background information on MPEG-2 and transcoding schemes is given in section 2. This section describes the MPEG-2 stream structure, different transcoding schemes and a basic transcoding architecture for bit rate reduction. The implementation details of the transcoder are given in Section 3. this section explains the block diagram of the transcoder that has been implemented and describes the alterations made to the input stream. Section 4 describes the performance analysis of the transcoder. The performance has been analyzed with respect to the reduction in bit rate, frame processing times and quality. An application of the transcoder as a bandwidth manager has been simulated in section 5 along with its performance analysis. Section 6 concludes the report with directions for future work and scope for improvements.

CHAPTER 2

BACKGROUND INFORMATION

This chapter gives some background information on the MPEG-2 standard, structure of an MPEG-2 bit stream, various transcoder architectures and transcoding modes. Some related work on transcoders is also described here.

2.1 MPEG-2

2.1.1 The Standard

MPEG-2 is an extension of the MPEG-1 international standard for digital compression of audio and video signals. MPEG-1 was designed to code progressively scanned video at bit rates up to about 1.5 Mbps for applications such as CD-I (Compact Disc Interactive). MPEG-2 is directed towards broadcast formats for higher data rates. It provides extra algorithmic 'tools' for efficiently coding interlaced video, supports a wide range of bit rates and provides for multi-channel surround sound coding. MPEG-2 achieves compression by using spatial and temporal compression algorithms. Spatial compression is achieved by removing redundancy within a video frame and temporal compression is achieved by difference encoding frames. For reducing redundancy within a video frame, MPEG-2 uses DCT based compression. To exploit temporal redundancy

in a video sequence the concept of motion compensation is used. It uses the fact that information between successive frames may not change a lot. Hence only the differences between successive frames are encoded along with some information about the motion vectors needed to reconstruct the picture.

2.1.2 Picture Types

In MPEG-2, three picture types are defined. The picture type defines which prediction modes may be used to code each block.

- Intra picture (I-picture) is coded with information from itself. Moderate compression is achieved by reducing spatial redundancy, but not temporal redundancy.
- Predictive picture (P-picture) is a picture which is coded using motion compensated prediction from a past reference frame or a past reference field.
- Bidirectionally-predictive picture (B-picture) is a picture which is coded using motion compensated prediction from a past and/or future reference frames.

The ‘display’ order is different from the ‘bit-stream’ order since all the reference frames needed to decode a frame must be transmitted before the dependent frame. For a given decoded picture quality, coding using each picture type produces a different number of bits. Typically I-frames are at least twice the size of a B-frame.

2.1.3 The Architecture

Figure 2 explains the MPEG-2 encoding and decoding process. The encoder subtracts the motion-compensated prediction from the source picture to form a 'prediction

error' picture. The prediction error is transformed with the DCT, the coefficients are quantized and these quantized values are coded using a VLC. The coded luminance and chrominance prediction error is combined with 'side information' required by the decoder, such as motion vectors and synchronizing information, and formed into a bitstream for transmission. In the decoder, the quantized DCT coefficients are reconstructed and inverse transformed to produce the prediction error. This is added to the motion-compensated prediction generated from previously decoded pictures to produce the decoded output.

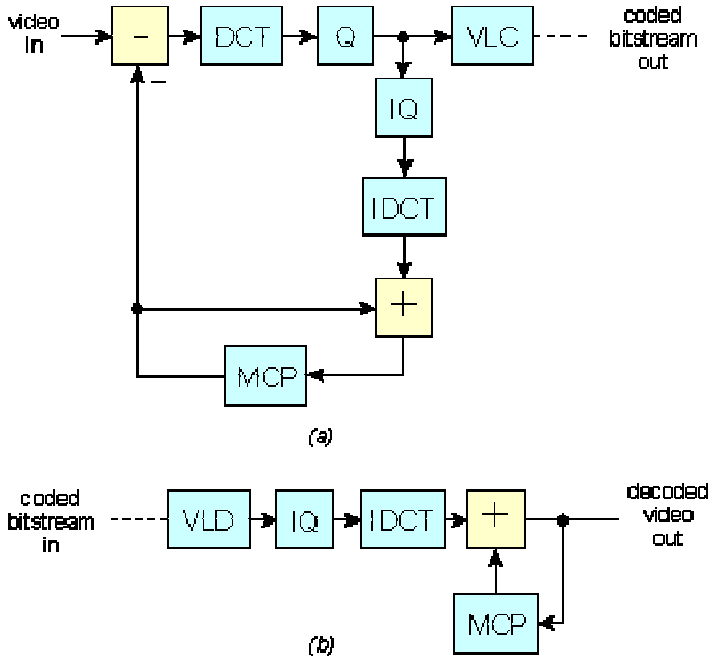

Figure 2.1 : MPEG-2 Encode – Decode architecture

2.1.4 The Stream Structure

The following figure shows the structure of an MPEG stream. An MPEG stream is a series of one or more video sequences. Each sequence is composed of many groups of pictures. Each group has one or more pictures / frames. A GOP allows the user to group together an arbitrary number of frames starting with an I-frame. The number of pictures in a GOP is not fixed by the standard. Each picture is a series of slices. Each slice is composed of a certain number of macroblocks. This number remains the same for the entire sequence and is contained in the sequence header. Each macroblock is made up of 6,8 or 12 blocks of DCT coefficients, depending on the number of luminance and chrominance components included.

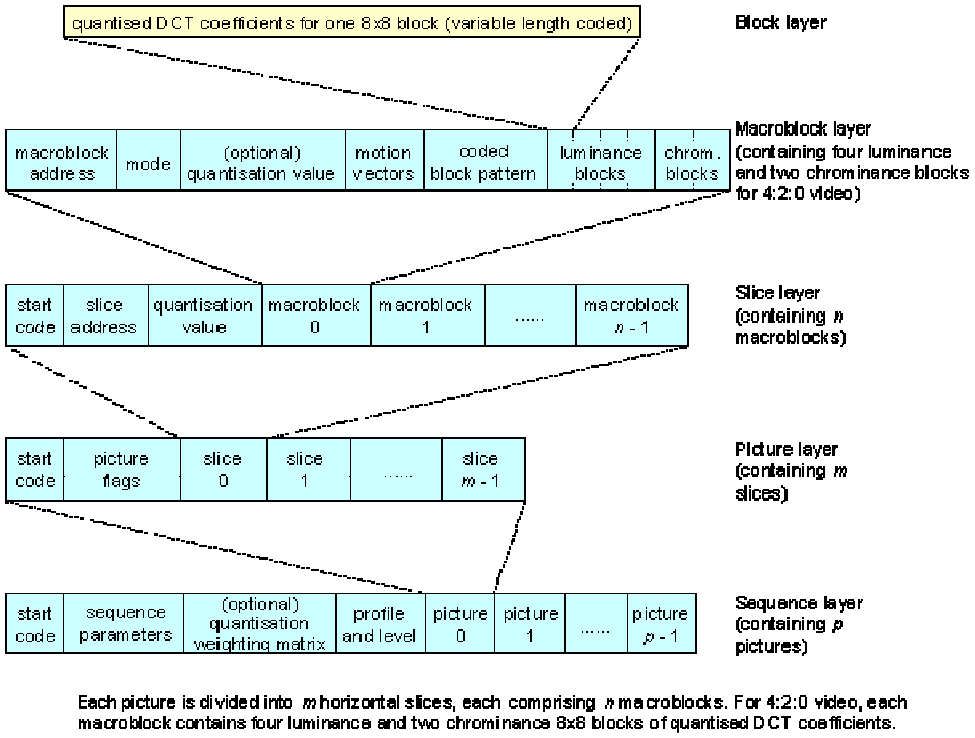

Figure 2.2 : The MPEG-2 Stream structure

A macroblock may be skipped from a stream if all the block coefficients are 0. So all the information in a macroblock has to be saved until it is known whether that macrolock will be output back on to the stream. This includes the motion vectors in the macroblock, if any. In the MPEG-2 codec, the motion-compensated predictor shown in Figure 2 supports many methods for generating a prediction. For example, the block may be 'forward predicted' from a previous picture, 'backward predicted' from a future picture, or 'bi-directional predicted' by averaging a forward and backward prediction. The method used to predict the block may change from one block to the next. Additionally, the two fields (namely field encoded or frame encoded) within a block may be predicted separately with their own motion vector, or together using a common motion vector. For each block to be coded, the encoder chooses between these prediction modes, trying to maximize the decoded picture quality within the constraints of the bit rate. The choice of prediction mode is transmitted to the decoder, with the prediction error, so that it may regenerate the correct prediction.

2.2 The Video Transcoder

2.2.1 Basic Architecture

The most basic configuration of a system including a transcoder is shown in Figure 4 and can be described as follows: an encoder compressing the incoming video signal at a bit rate of Z_1 kbits/s, then a transcoder where video is converted from Z_1 kbits/sec to Z_2 kbits/sec and finally a decoder which displays the resulting video sequence to the user.

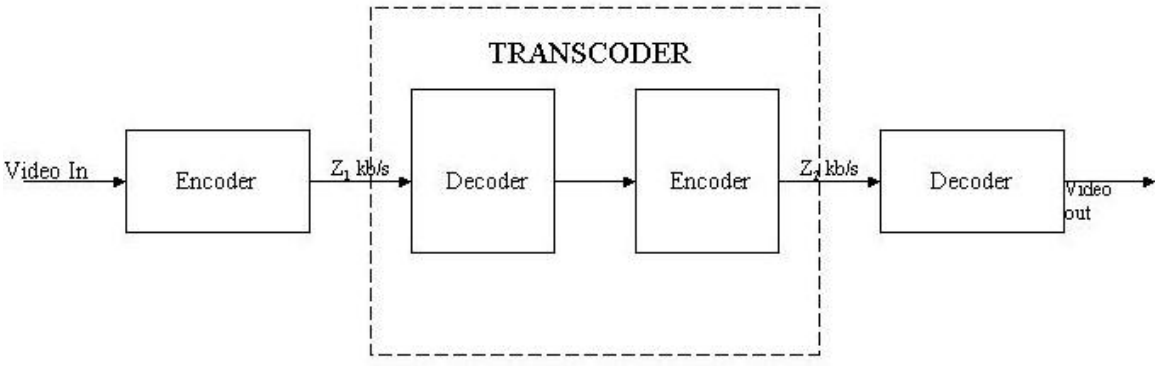

Figure 2.3: Basic Transcoder Architecture

As shown in Figure 3.3, a transcoder consists of a cascaded decoder and then encoder and a decoder . The case of interest in this thesis is one where $Z_2 < Z_1$ kbits/s. The incoming bit stream is first run through a VLD which gives the value of the quantized coefficients and motion vectors. Next these coefficients are dequantized and inverse transformed by an IDCT. These operations yield the signal ΔX_n for the picture X_n. Next the picture is reconstructed by adding the prediction from the previously decoded picture X_{n-1} . After decoding the incoming signal the encoder follows. As seen in figure 3.4, first the prediction P_n is subtracted from the picture X_n which gives the residual signal ΔY_n. This signal is transformed using a DCT and further quantized (Q_2). The quantized DCT coefficients are put through a variable length decoder and then output as a bit stream. Since the previously quantized image needs to be stored quantized DCT coefficients are again dequantized and are further inverse transformed by an IDCT to yield the residual reconstructed picture $\Delta Y_n + E_n$ where E_n represents the quantization error. Further more the picture Y_n is reconstructed by adding the same prediction P_n that

was subtracted earlier in the process. Besides the operations of DCT and IDCT, calculation of P_n involves a computationally expensive motion estimation operation.

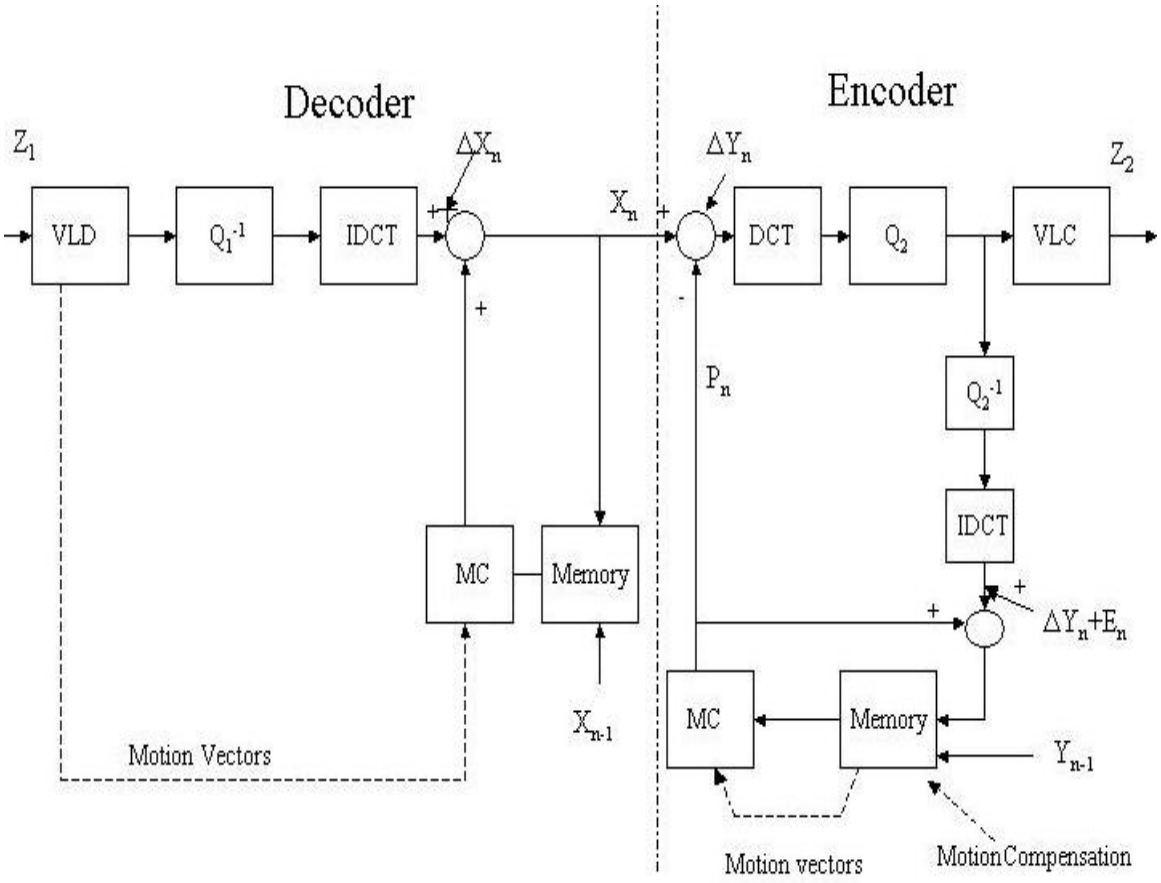

Figure 2.4: Block diagram of a detailed transcoder

2.2.2 Reducing Complexity of The Model

The following techniques can be employed to reduce the complexity of the model

- Passing the motion vectors directly from the decoder to the encoder therefore avoiding the computationally complex task of motion estimation in the encoder of the transcoder.
- Macroblock and block information can be passed directly.
- Use of simple requantization transcoding, where motion vectors and macroblock are passed directly.
- More advanced methods which reduce memory requirements and DCT/IDCT operations.

2.3 Modes of Transcoding

Three different modes of transcoding architecture are possible.

2.3.1 Rate Reduction

Rate based reduction implies that Z_2 is less than Z_1. A simple way to do that is to increase the quantization step when encoding the stream in the transcoder. This decreases the number of non zero quantized coefficients thus reducing the number of bits in the outgoing bit stream.

2.3.2 Temporal Transcoding

In temporal transcoding, the bit rate reduction is achieved by dropping some frames in incoming bit-stream. In many cases the same frame rate as the incoming produces unacceptable video quality when transcoding is too high. Thus dropping of frames is required to maintain end to end delay requirement. When the frame dropping occurs, depending on the type of frame dropped the motion vectors may need to be recomputed requiring a costly motion estimation operation.

2.3.3 Resolution Reduction

Decreasing the size of the frames transmitted will reduce the amount of information needed to be transmitted from the source to the destination. To do this a down sampling filter can be inserted in the transcoder between the decoder and encoder. After the stream has been decoded completely the down sampling filter is applied and then encoded again.

2.4 Concerns with MPEG-2 Streaming

In a highly heterogeneous network environment, the transfer of an MPEG-2 stream is tough to manage. The bandwidth along different links varies over time. So, if a high bit rate MPEG stream suddenly comes across a low bandwidth link, it will drastically decrease the frame rate. At a gateway or proxy, it will eventually cause dropping of packets. A random loss of information will distort the MPEG stream. A dynamic bit rate adaptation at real time would be useful, albeit at the loss of quality.

Moreover, if it is a low bandwidth link, or if the devices connected are laptops and palm devices, which do not have the processing power to handle a high quality stream, a higher frame rate and a low bit rate is required.

Using the enhancement layers in MPEG-2, only 2 discrete levels of quality and bit rate can be achieved. An MPEG-2 transcoder can provide flexible adaptation of the bit rate to the available bandwidth. This transcoder can be placed at a gateway in the end system, which will dynamically adapt to the incoming MPEG-2 stream.

2.5 Related Work

Building adaptive video gateways using transcoders has been an interesting research area in multimedia networks, for a long time. This section briefly describes some of them.

2.5.1 An Application Level Video Gateway

An application level video gateway was developed in University of California at Berkley so that a multicast video transmission can be decomposed into multiple sessions with different bandwidth requirements. The video gateway transparently connects pairs of sessions into a single logical conference by performing bandwidth adaptation using transcoding and rate control. The transcoder converts motion-JPEG streams to H.261, considering the low bit rate requirements of H.261 compared to Motion-JPEG. [5]

2.5.2 Video Transcoding for Universal Multimedia Access

A basic video transcoding model for H.263 streams, using rate reduction and resolution reduction, was developed in Ericsson radio systems. The transcoder was used for adapting video streams to different types of terminals with different terminal capabilities such as screen size, amount of available memory, processing power and type of network access. This effort was made to serve the Universal Multimedia Application related to MPEG-7 standardization activities. The UMA aims in enabling the client devices to access rich multimedia content and consume it in different modalities, than in which the original information has been captured. [4]

2.5.3 An MPEG-1 Transcoder

An MPEG-1 audio and video transcoder has been designed in the C&C Research lab. The performance of this transcoder depends on the processing capabilities, network bandwidth, transport protocol and the code optimization. The implementation was undertaken as a sample application for the project of media processing in networks. The team is now working on developing a framework, which will allow multiple streams to flow through a node, some of which may get transcoded at the node. [3]

2.5.4 Reducing Drift Errors

The motion vectors are not re-computed for any of the macroblocks. When a macroblock is skipped, the previous motion vectors are reset. The motion vectors in the input stream were based on the earlier coefficients. Hence the frames regenerated using the motion vectors in the output stream will not exactly match the original frames. This is

called as the drift error, which is reset only when an I-frame is encountered again. Youn and Sun have designed a scheme to refine the motion vectors being re-used, in order to counter the drift error. A new motion estimation is performed using a fast-search adaptive motion vector refinement scheme. [1]

CHAPTER 3

DESIGN AND IMPLEMENTATION OF THE TRANSCODER

Transcoding of input bit stream has been achieved using two methods.

1. Increasing quantization scale
2. Dropping of intermediate frames.

This section describes the design and implementation of the 2 methods.

3.1 Increasing Quantization Scale

A fast DCT domain transcoder has been designed for the purpose of re-quantizing the stream. The incoming pre-encoded video stream is first entropy decoded. Entropy decoding here is a 2 stage process: Variable length decoding and Inverse scan. Variable length decoding is basically a Huffman decoding process. The inverse scan is a way in which one dimensional data recovered using the Huffman decoding process is converted into a two dimensional array of coefficients. Motion vectors, some header information, macroblock information and blocks of DCT coefficients are all entropy encoded in the stream.

The DCT-ized blocks are then inverse-quantized before requantizing to a new scale. The newly quantized blocks are then entropy encoded before putting them back on the stream.

3.2 Block Diagram

This figure shows the functional block diagram for the transcoder, which has been designed.

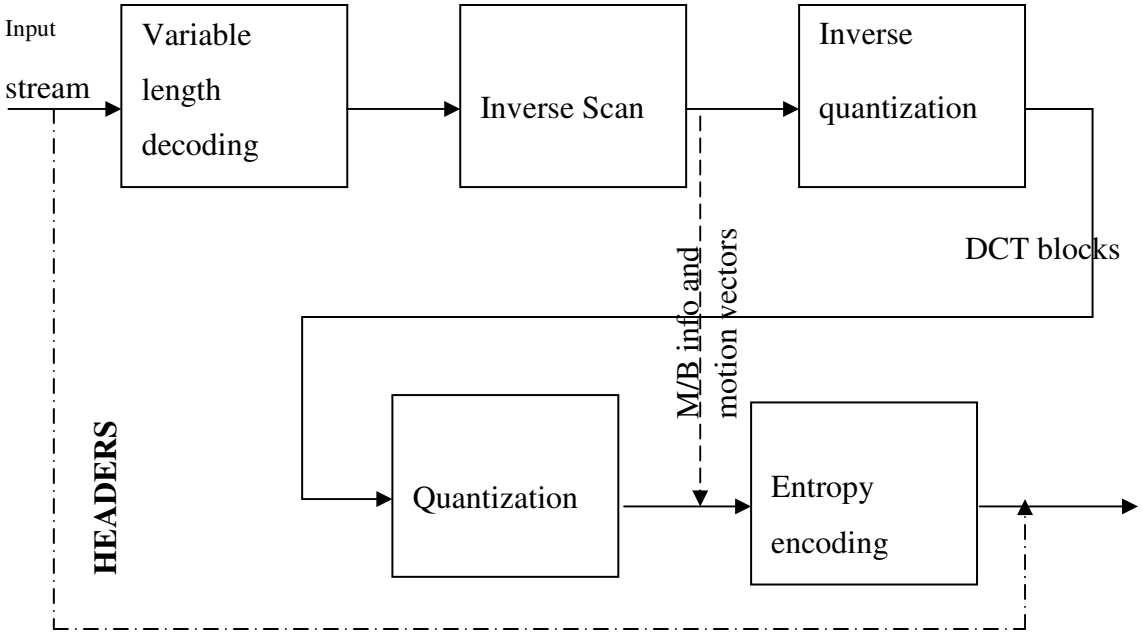

Figure 3.1 : Block diagram of the transcoder

Until the first picture header is encountered, the contents of the input stream can be just copied on to the output, without any changes. The individual blocks of a macroblock are subject to inverse quantization and then requantization. However, these

blocks cannot just be parsed from the input, munched and then sent back to the output, for the following reasons.

1. A count has to be maintained of the number of macroblocks skipped within a slice as this number has to be output in the next non-skipped macroblock. The first and the last macroblock of any slice may not be skipped. A macroblock cannot be skipped in an I-frame. In a B-frame, a macroblock following an intra coded macroblock cannot be skipped.
2. The slice header has information on the quantization scale for all the macroblocks in a stream, unless a macroblock in that slice carries a different quantization scale. If a macroblock carries a different quantization scale than its previous macroblock, it has to be reflected in the mcroblock type. This information is known only at the time of quantizing the stream.

Following the slice header, a variable number of macroblocks appear in the stream. Each macroblock comprises the following information. These items have to be saved temporarily until the coefficients have been re-quantized and it is known whether the macroblock will be skipped or included in the output stream. Some information changes after the re-quantizing process.

The temporary storage involved for the transcoder is only the space required to store these items. After the new macroblock is generated, it is output back on to the stream.

Item	Description	comments
Address Increment	Number of macroblocks skipped in the stream before the current macroblock. Has to be value 1 in the first macroblock	The new address increment is entropy encoded and put back in to the stream at the next non-skipped macroblock.
Macroblock type	This gives information on one or more of the following. • Whether it contains a new quantization scale, • whether it is intra coded, • whether it is forward encoded, • whether it is backward encoded • whether it is pattern encoded	When a macroblock has a different quantization scale than the previous macroblock, the type bit is modified to indicate this. The quantization scale is transmitted later in the macroblock. An initially pattern encoded macroblock can now be all zero. It cannot be skipped if it the first or last macroblock of a slice. But it is not pattern encoded anymore. The macroblock type has to be modified to indicate this.
Motion type	This gives information on the macroblock prediction type	The value is the same as in the input stream
DCT type	Flag indicating whether the macroblock is frame DCT encoded or field DCT encoded	The value is the same as in the input stream
Quantization scale	If the quantization scale for this macroblock is different than the previous, the new quantization scale is included in the macroblock.	An indication that this information is present, comes from the macroblock type.
Motion vectors	If the macroblock type specifies forward or backward or both, motion vectors are carried in the macroblock	The value is the same as in the input stream
Coded Block pattern	This is a variable length code which indicates which of the blocks in a macroblock are non zero. For an intra coded macroblock, all the blocks are non zero.	The value changes for a non-intra coded macroblock. If all the blocks are 0, coded block pattern would be zero, indicating that the macroblock is to be skipped.
Block coefficients	The DCT coefficients after being entropy encoded are transmitted next.	These are the coefficients which are re-quantised. Hence the value changes.

Figure 3.2 : Processing of Macroblock information

3.3 Design Issues

1. The values of the quantization factor is limited by a non linear scale varying from 1-56 or 1-112, depending on the quantization type of the stream. This puts a limit on the quantization achieved. In a high quality MPEG-2 stream, the quantization scale would be low. Hence, higher compression can be achieved after transcoding.
2. The motion vectors are not re-computed for any of the macroblocks. When a macroblock is skipped, the previous motion vectors are reset. The motion vectors in the input stream were based on the earlier coefficients. Hence the frames regenerated using the motion vectors in the output stream will not exactly match the original frames. This is called as the drift error, which is reset only when an I-frame is encountered again. Thus, the quality of the stream is lesser than a re-encoded stream with higher quantization. However, in the process of re-encoding, the entire objective of transcoding would be lost because of the enormous processing complexity and slow speed. Hence the compromise is made.

3.4 B-frame Dropping

When the picture header is parsed, if it is observed that it is a B-frame, the frame is not output. The implementation of a B-frame dropper is very trivial. Here also, the motion vectors are not changed.

The B-frame dropper is more useful as a buffer controller. If the buffer at a proxy tends to overflow, the B-frame dropper can be employed, instead of arbitrarily dropping packets.

CHAPTER 4

PERFORMANCE ANALYSIS OF THE MPEG-2 TRANSCODER

The performance analysis of the MPEG-2 transcoder has been divided into 3 parts as follows:

1. Frame processing times
2. Frame sizes
3. Quality

This chapter discusses these results. The transcoder was run on a number of mpeg streams. The results here are discussed based on

1. 20 minute stream NBA.mpg which is made up of 352x480 frames
2. 2 minute stream tennis.mpg which is made up of 352x240 sized frames.

4.1 Reduction in Frame Sizes

The following graph summarizes the frame sizes for the NBA.mpg file, under different transcoding rates. The frame sizes reduce drastically from scale 1 to scale 1.5 and 2, but later slows down. The stream itself was of a low quality. The maximum I-

frame size is only 23000 bytes. Hence, the effects of transcoding after scale 7 are negligible.

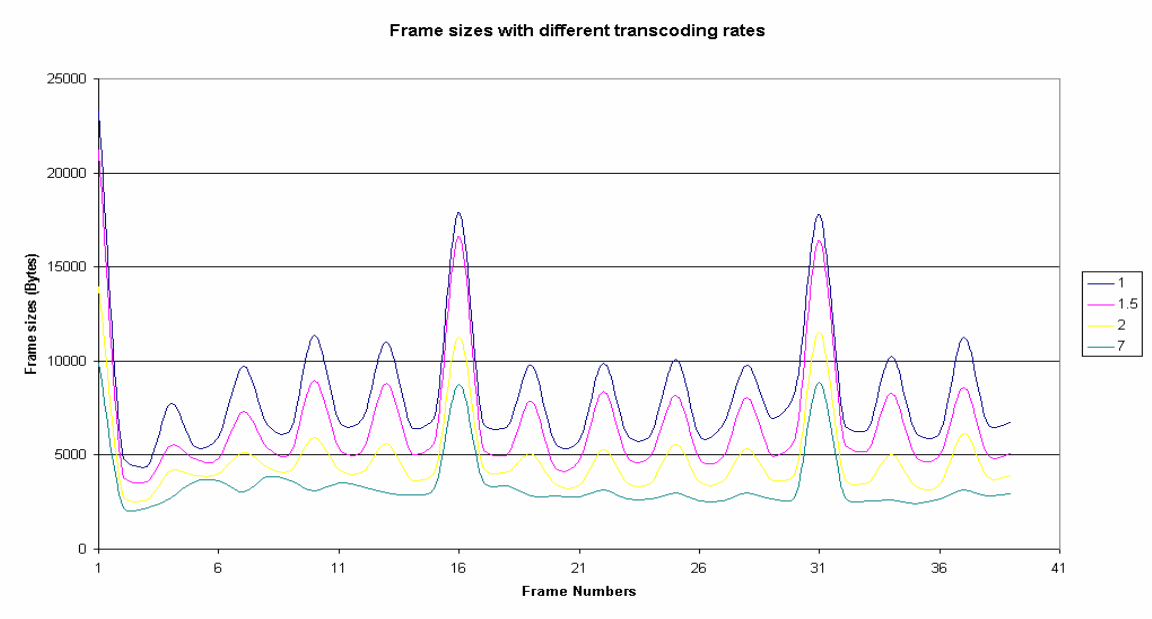

Figure 4.1 : Frame sizes with different transcoding rates for NBA.mpg

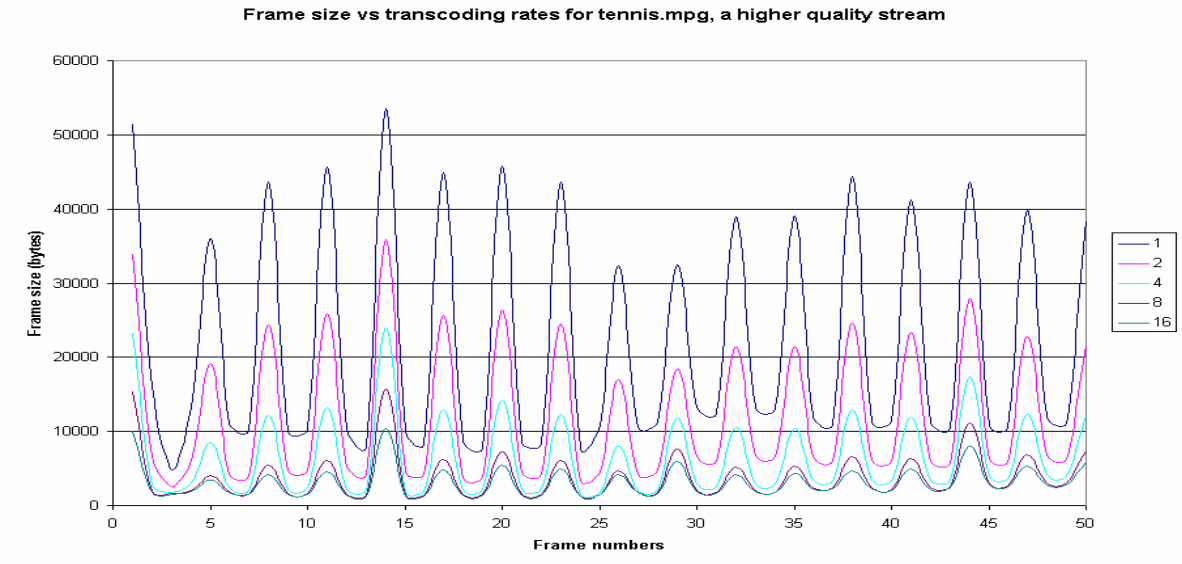

Figure 4.2: Frame sizes with different transcoding rates for tennis.mpg

With *tennis.mpg*, there is a significant decrease in frame sizes up to a transcoding rate of 8. then it dwindles. However, for I frames, there is a significant drop in size even at a rate of 16. the frame size falls almost 6 times at that rate. The maximum I frame size for tennis.mpg is 55000 bytes showing that, this a higher quality stream.

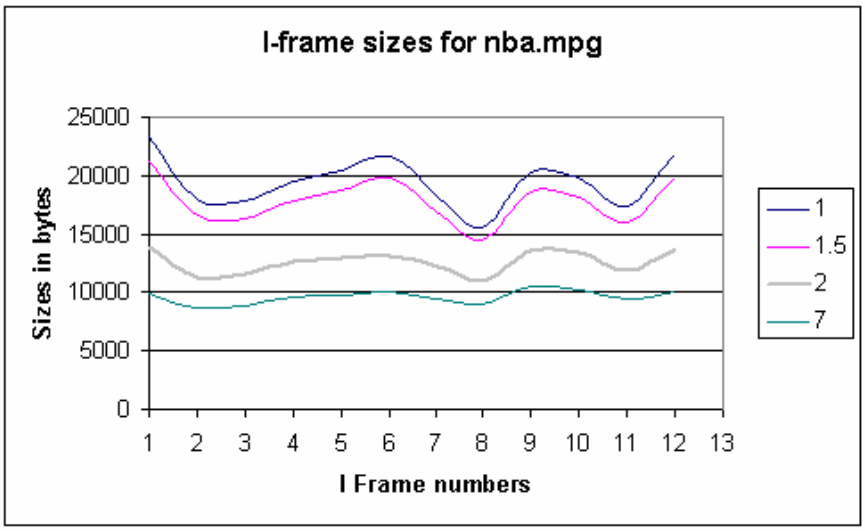

Figure 4.3: I-frame sizes for NBA.mpg

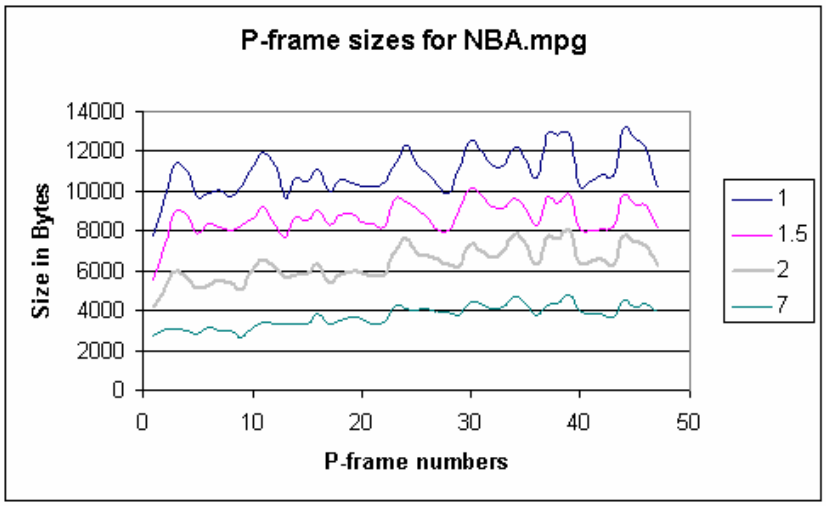

Figure 4.4: P-frame sizes for NBA.mpg

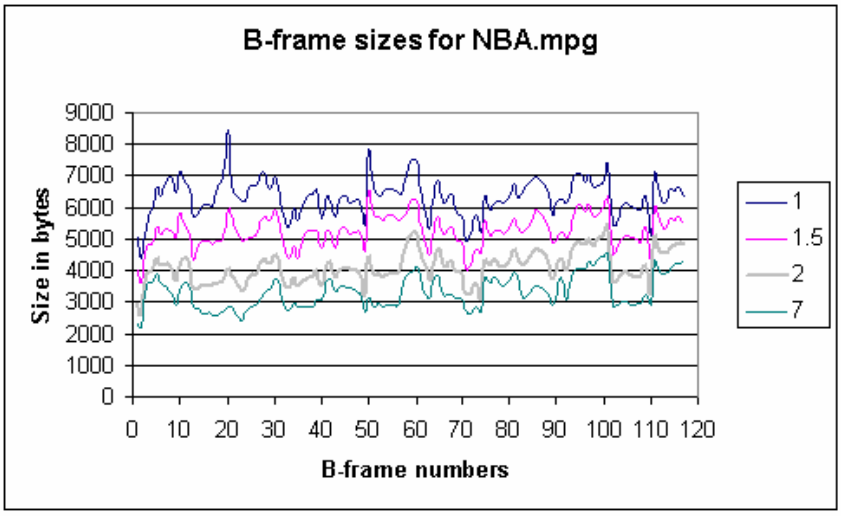

Figure 4.5: B-frame sizes for NBA.mpg

The above graphs show the reduction in frame sizes, separately for I, P and B pictures. The reduction ratio is pretty much a constant for I, P and B pictures. The following graphs show the I,P and B-frame size comparisons for tennis.mpg.

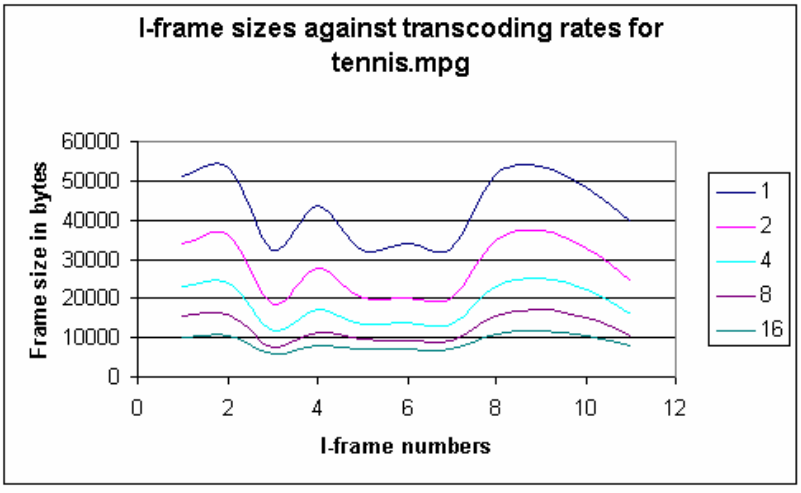

Figure 4.6: I –frame sizes for tennis.mpg

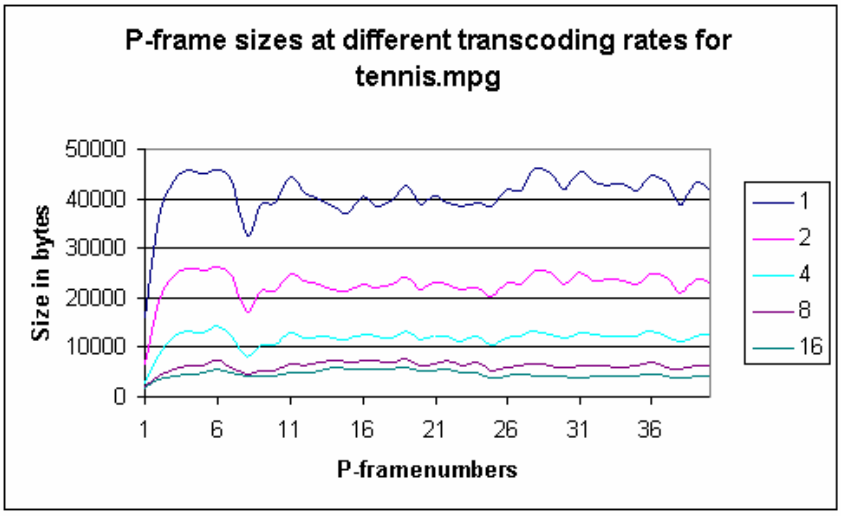

Figure 4.7: P-frame sizes for tennis.mpg

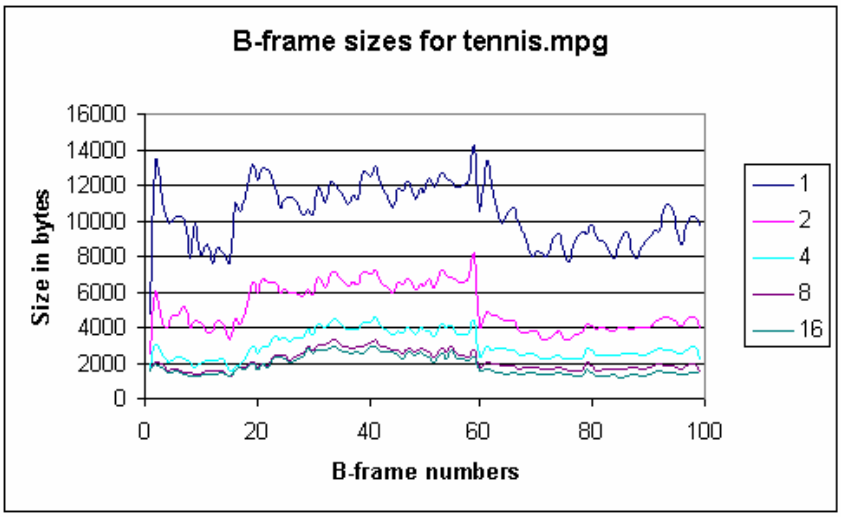

Figure 4.8: B-frame sizes for tennis.mpg

The above graphs show that for tennis.mpg, the I frame sizes decrease drastically even at a transcoding rate of 16. But the P and B frame size reduction attains a saturation at rate 8.

4.1.1 Comparison of Bit Rates

The following graphs show the comparison of bit rates at different transcoding scales for tennis.mpg and NBA.mpg

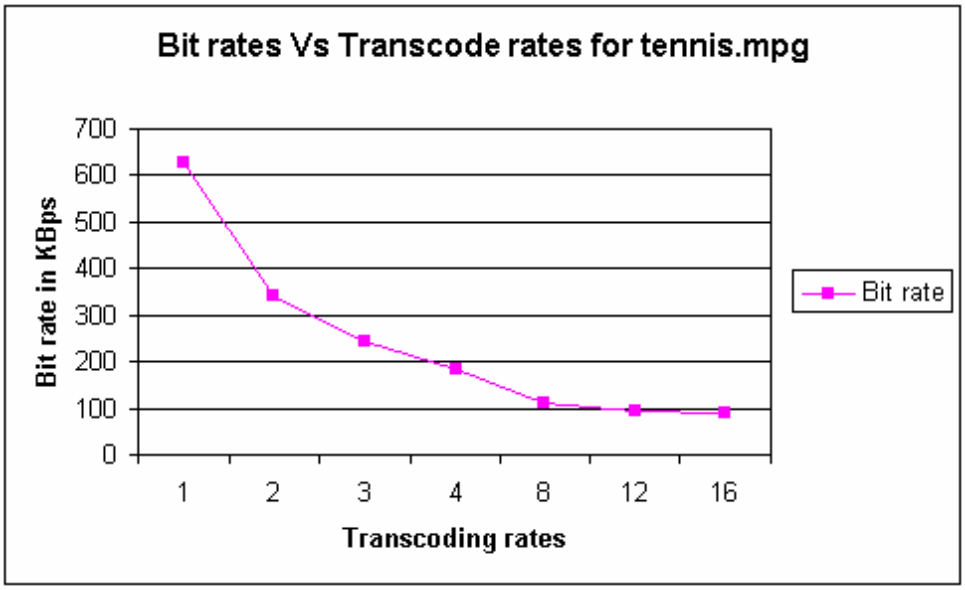

Figure 4.9 : Comparison of Bit rates in KBps at different transcoding levels for tennis.mpg

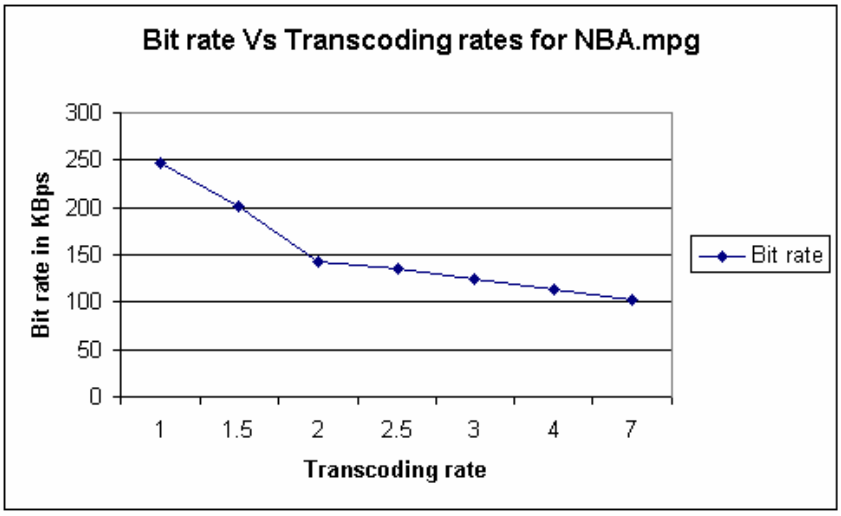

Figure 4.10 : Comparison of Bt rate in KBps at different transcoding levels for NBA.mpg

The speed of the transcoder is crucial to its performance. The speed is highly processor dependent. The transcoder was tested for speed on an UNIX machine. The following figure shows the frame processing times for NBA.mpg.

The results also show that at higher transcoding rates, the frame processing time reduces slightly.

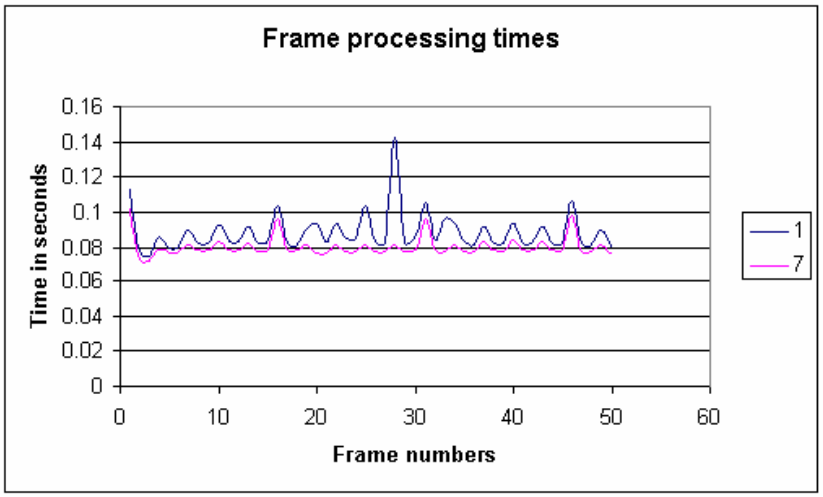

Figure 4.11: Frame processing times for NBA.mpg

The frame processing time also depends on the size of the picture. The following graph shows the results with tennis.mpg. the pictures are smaller in this case. So the frame processing times decrease. But it is slightly compensated by the fact that they are better quality and hence larger in bit size. It was also observed that at very high transcoding rates, the frame processing time tends to increase slightly, because for each macroblock, the highest quantization scale just below the maximum level has to be found.

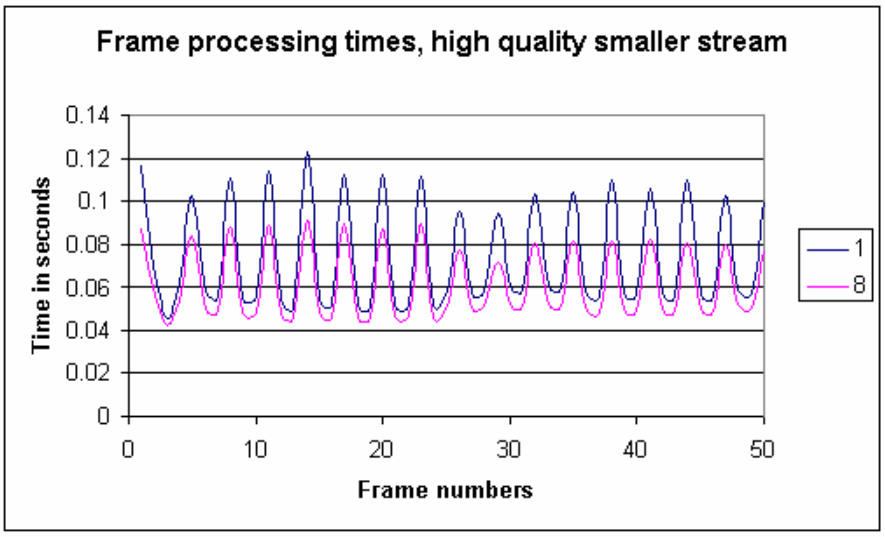

Figure 4.12: Frame processing times for tennis.mpg

The average frames per second output for both files are tabulated below.

	Transcoding rate 1	**Transcoding rate 8**
NBA.mpg (average)	11.3	12.5
NBA.mpg (I – only)	9.4	10.2
NBA.mpg (P-only)	10.7	12
NBA.mpg (B-only)	12	13
Tennis.mpg	13.7	16.5
Tennis.mpg (I –only)	9.5	12.5
Tennis.mpg (P –only)	9.5	12.6
Tennis.mpg (B – only)	18.2	20.8

Figure 4.13: The average frames per second generated by transcoder for the 2 streams

	Using Transcoder	**Re-encoding at different bit rate**
Tennis.mpg (150 frames)	10 seconds	26 Seconds
NBA.mpg (150 frames)	9 seconds	25 seconds

Figure 4.14: Comparison of times taken to transcode and completely re-encode

The above table shows the advantage of transcoding over completely re-encoding the stream. It also shows that transcoding is around 2.5 times faster than completely re-encoding the stream. Hence, on systems in which faster MPEG encoding can be performed, transcoding can also be achieved faster.

4.3 Quality

The following pictures show a comparison of picture quality at different transcoding levels.

Figure 4.15: An I frame from tennis.mp,g no transcoding

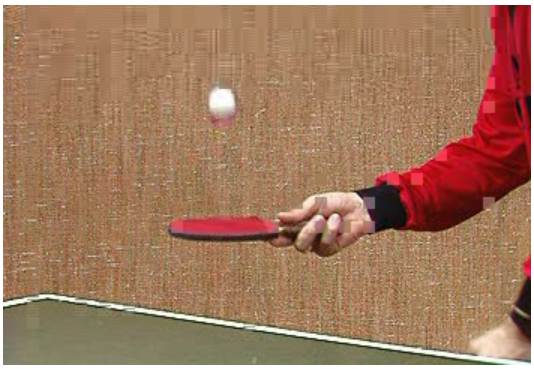

Figure 4.16: An I frame from tennis.mpg, transcode rate = 2

Figure 4.17: An I frame from tennis.mpg, transcoding rate = 8

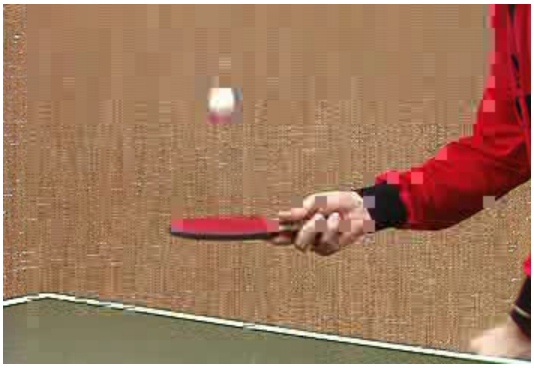

Figure 4.18: An I frame from tennis.mpg, transcoding rate = 16

Figure 4.19: A B frame from tennis.mpg, no transcoding

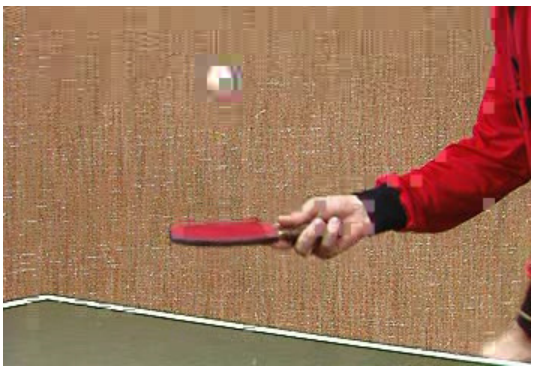

Figure 4.20: A B frame from tennis.mpg, transcoding rate = 4

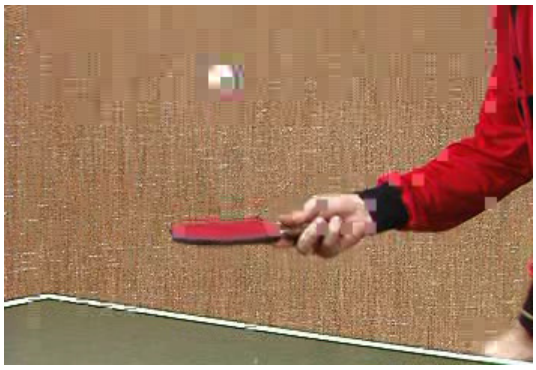

Figure 4.21 : A B frame from tennis.mpg, transcoding rate = 8

Figure 4.22: A B frame from tennis.mpg, transcoding rate = 16

4.4 Summary of Findings

Based on the observations above, we can see that the transcoder is a useful tool to dynamically reduce the bit rate of an MPEG-2 stream. The bit rate of the stream reduces by a factor of about 7, after which saturation is reached. Its speed is highly dependent on the processing environment. It is about 2.5 times faster when compared to re-encoding the stream, giving an indication of its performance. The quality of P and B frames deteriorate more compared to I frames. The drift error described earlier also plays a role in this. By incorporating drift free architectures, this quality can be improved. The use of the MPEG2 transcoder in bandwidth management is explained in the next chapter.

CHAPTER 5

THE ADAPTIVE BANDWIDTH MANAGER : AN APPLICATION

This chapter describes the design and implementation of the bandwidth manager. First, the architecture of the bandwidth manager is discussed. The bandwidth manager uses the transcoder and frame dropper to adapt the bit rate of the stream. This is followed by the performance analysis of the adaptive transcoding system.

5.1 The Bandwidth Manager

The bandwidth manager can be placed at the proxy or at the client. When placed at the proxy, it would be close to the transcoder and feedback is faster. The design of both these architectures are discussed here. The performance analysis and the relative advantages of these schemes are discussed later.

5.1.1 Proxy Side Placement

As shown in Figure the adaptive bandwidth management system resides at the proxy between the video server and the low bandwidth device. This is an example of a streamed transcoding proxy wherein the transcoder can output partial frames hence

improving response time. The adaptive management system adapts the incoming video stream to the bandwidth it observes between the proxy and the client. The system is being fed data at Z_1 kb/s and data is being removed at Z_2 kb/s. Since $Z_2 << Z_1$ the system will transcode and drop frames if required from the incoming stream and buffer whatever data cannot be sent at the present time. To ensure smooth video and avoid an underflow or an overflow, the system has two defined thresholds in the buffer between in which it tries to maintain the data. MPEG-2 stream by nature is very bursty due to unequal frame sizes and hence requires the definition of an overflow and an underflow threshold to maintain, to smoothen delivery of video.

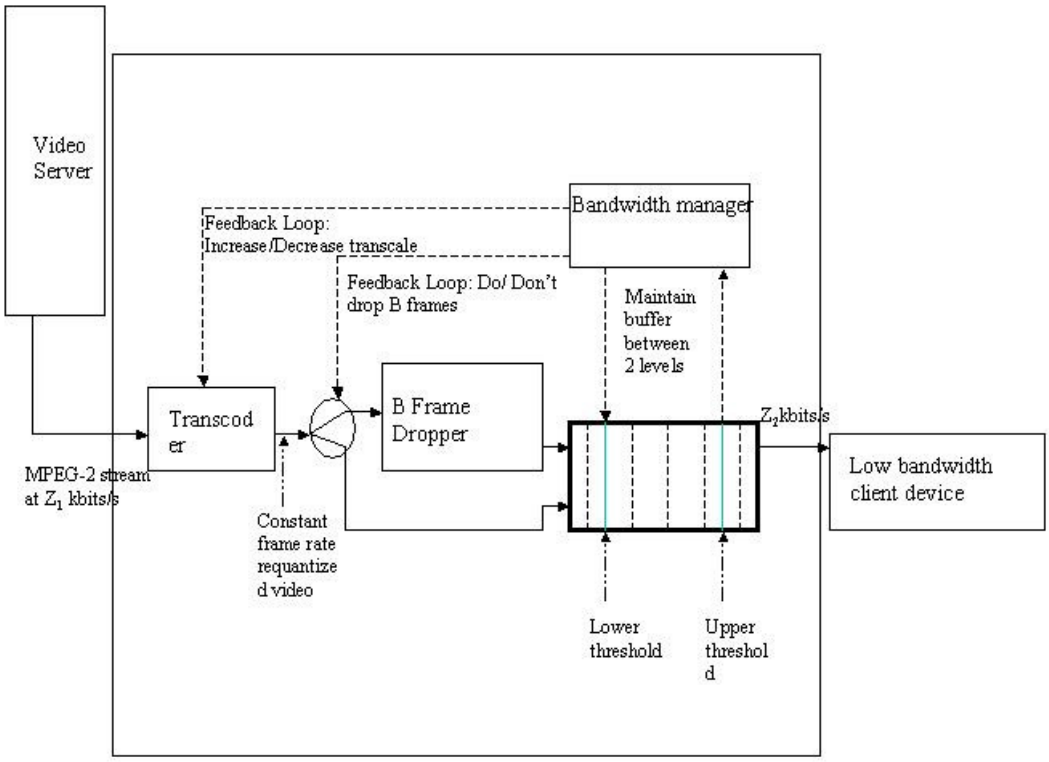

Figure 5.1 : Architecture for Proxy side bandwidth manager

The system consists of the following components:

1. **Bandwidth Manager**

   The bandwidth manager monitors the buffer and the output bandwidth Z_2 to ensure that the buffer is filled between the two levels. To do this it sends feedback to the rate based transcoder to change the quantization scale. The interval to send feedback can be changed. The current implementation sends feedback every one second. If the buffer occupancy has crossed the higher threshold then it asks the transcoder to increase the quantization and if it crosses the lower threshold it asks the transcoder to decrease the quantization. If the buffer overflows at some point of time the bandwidth manager will pass the output of the transcoder through the B frame dropper which will drop enough frames so that the buffer is restored to its stable state.

2. **Transcoder**

   A rate based transcoder lowers the bit rate of the incoming bit stream by a simple re-quantization of the DCT coefficients. Even with the increase in the quantization level the rate based transcoder still maintains the same frames/sec thus sending less data per second to represent the same number of frames and hence saving on bandwidth.

3. **B frame Dropper (Temporal Transcoder)**

   The bandwidth manager monitors the buffer and an estimated overflow asks the B frame dropper to parse the incoming transcoded bit stream and drop the B frames from it. Thus overflows are prevented. P frames depend on I frames to be

reconstructed. B frames depend on I and P frames to be reconstructed. Hence if I or P frames have to be dropped time has to be spent in expensive motion vector re-computation.

The objective of the above system is to avoid overflows and minimize underflows.

5.1.2 Client Side Management

When the bandwidth manager is placed at the client, it monitors the frames per second actually received over the network. Based on the client requirements, it send a feedback to the transcoder located at the source or at an intermediate proxy. The transcoder then reduces the bit rate and hence increases the frame rate. The B-frame dropper is not required in this case. The client can get the MPEG-2 stream at the quality and frame rate required.

A network of 60KBps bandwidth has been simulated. UDP is used to blast the stream on to the client. The streams are blasted in chunks of 20Kbytes. The bandwidth manager located at the client counts the number of frames received per second and provides feedback to the transcoder situated in the proxy. The feedback channel was tried using reliable TCP and unreliable UDP. With TCP, the response is always there, but with some lag. In UDP, sometimes the feedback is lost. But, on the whole it gives a better performance.

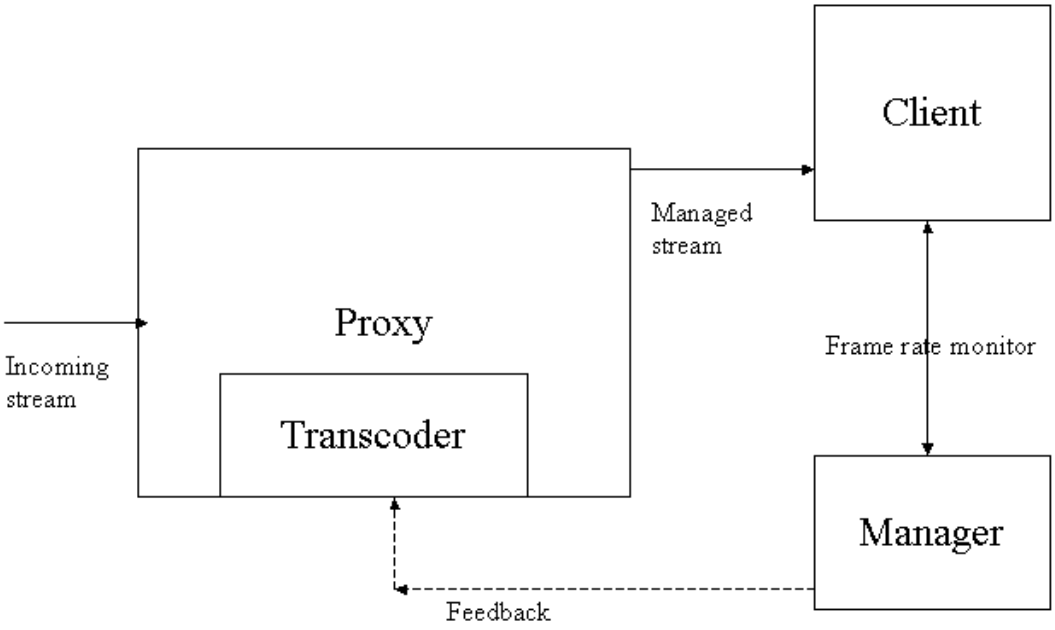

Figure 5.2 : Block diagram for client side bandwidth manager

5.2 Bandwidth Manager Performance Analysis

This chapter describes the performance analysis of the bandwidth manager, which employs the transcoder and the frame dropper to manage a low bandwidth link. As mentioned before, the bandwidth manager can be placed at the client side or along with the transcoder at the gateway. The results using both the methods are described here.

5.2.1 Manager at The Gateway

A bandwidth trace was created, which generates bandwidth between 50KBps and 70KBps, for the output link. The buffer requirements at the gateway to send the stream to the output link without an overflow were studied. A goal was set to keep the buffer below

100000 bytes. The following figure shows the buffer requirement without the transcoder. The buffer size required keeps increasing, thereby resulting in eventual loss of data.

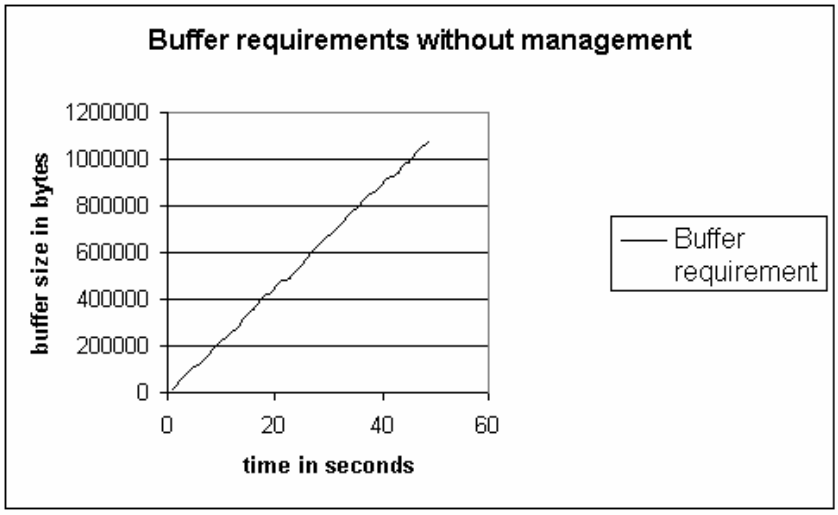

Figure 5.3 : Buffer size required, without any bandwidth manager employed

To maintain the buffer at less than 100000 bytes, 2 thresholds were maintained, 40000 and 70000 bytes. If the buffer required exceeded 70000, the quantization scale was increased by 2. However, if the buffer recorded an underflow, the quantization scale was decreased, but only in steps of one. This was done so that, a rapid increase of quality does not create a surge of data from the transcoder. As a means of precaution, when the buffer exceeded 80000, the quantization was increased by 4, since a little surge can cause overflow. The objective was to avoid an overflow and minimize underflow. As observed a buffersize never crossed 90000 bytes.

The following graph shows the buffer requirement, when the transcoder was employed to reduce the bit-rate of the stream. The next graph shows the frames per second output on the link and the transcoding rate required.

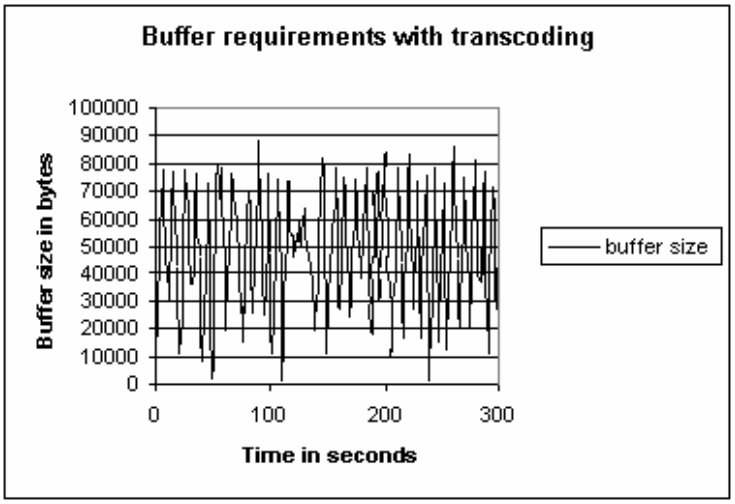

Figure 5.4 : Buffer variations with transcoder used.

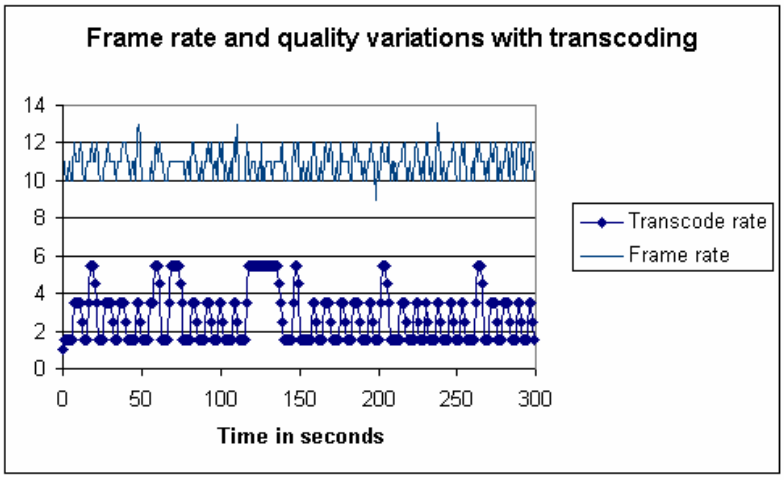

Figure 5.5 : Frame rate and quality variation with transcoder employed

The thresholds were found to be optimum, when they were kept at the maximum and minimum levels of bandwidth variation. If the higher threshold is kept very low, it will not give an optimum quality level. So, the idea was to get the higher threshold as high as possible, leaving some room for a surge of data. But, it is to be noted that a sudden surge of data, can overflow the buffer. If the buffer size is increased, we can increase the higher threshold by that much size.

In the above graphs, it is seen that the maximum transcoding rate required is 5.5. Some times, such a low quality may not be acceptable. Instead a lower frames per second may be okay. The following graph shows the frames per second on the output link with only the frame dropper. The next graph shows the buffer requirement for the same. Even though the effective fps is 4-8 fps buffer occupancy is very stable.

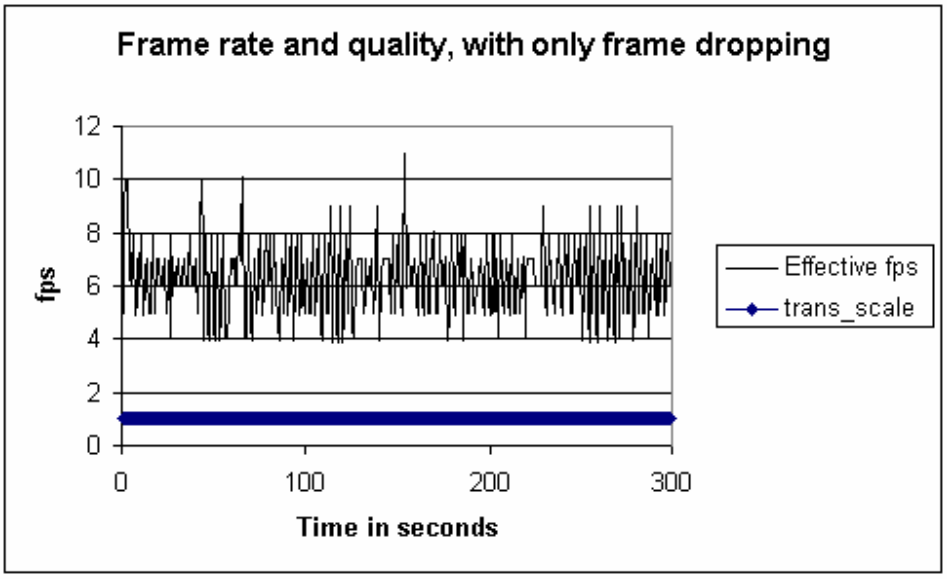

Figure 5.6 : Frame rate variations with only frame dropping

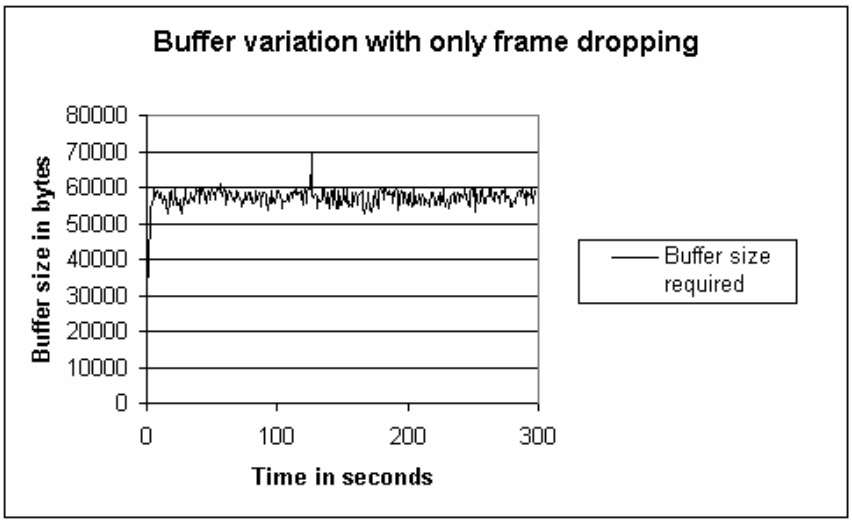

Figure 5.7: Buffer required with only frame dropping

In the above graphs, the quality remains good. But the frames per second varies a lot and may be very low. Besides that dropping only frames leads to a jerky display. A combination of the transcoder and frame dropper can give optimum results.

```
Every one second :
begin
if( UPPER_THRESHOLD < DATA_READ - BANDWIDTH + PREV_SIZE)
      if ( trans_scale < 3.5  )
                  trans_scale = trans_scale + 2 ;
else
      if ( trans_scale > 2 && LOWER_THRESH > DATA_READ - BANDWIDTH +
PREV_SIZE)
            trans_scale = trans_scale - 1 ;

if ( DATA_READ + PREV_SIZE > MAXBUFFER_SIZE )
      PREV_SIZE    =    SIZE_AFTER_DROPPING(DATA_READ+PREV_SIZE)    -
BANDWIDTH
else
      PREV_SIZE = DATA_READ-BANDWIDTH+PREV_SIZE;
end
```

Figure 5.8: Pseudo code for buffer management with transcoding and dropping

The following graphs show the buffer requirements and frame rate variations with both the transcoder and the frame dropper employed. The maximum buffer size required is around 60000 bytes. The transcoding rate required is only 2.5 for most of the time.

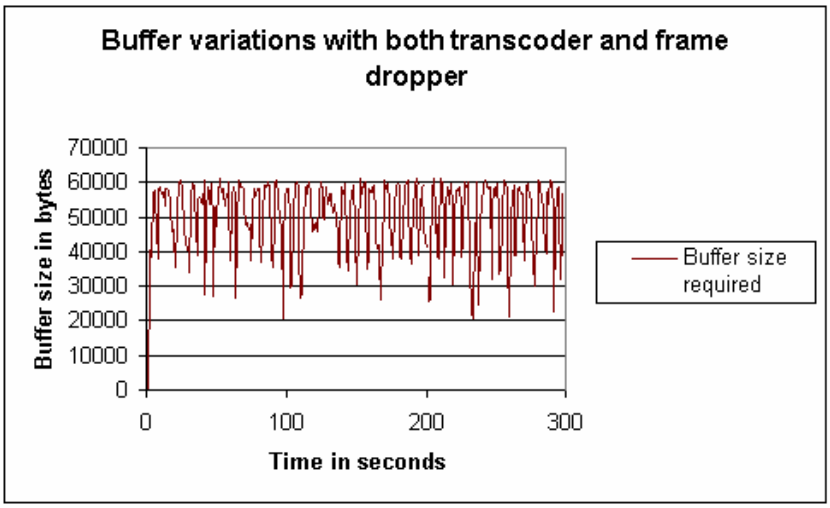

Figure 5.9: Buffer variations with transcoder and frame dropper

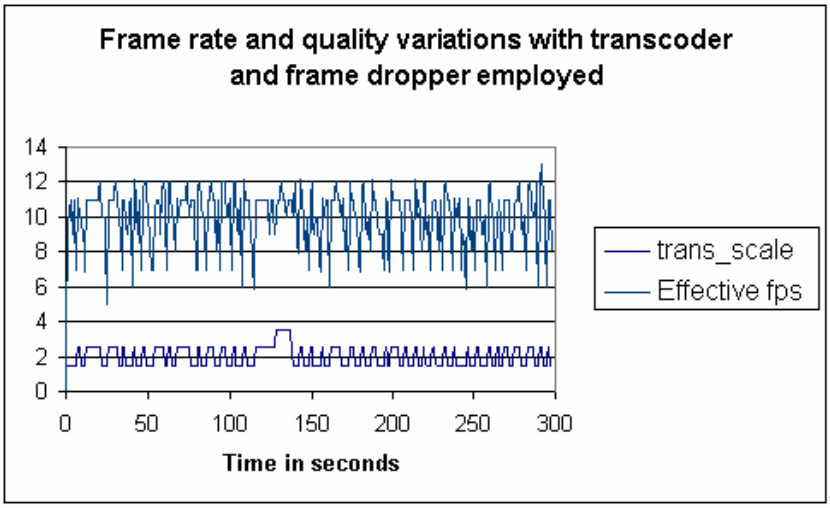

Figure 5.10:Frame rate and quality variations with transcoder and frame dropper

In these graphs, the bandwidth of the output link remains pretty much a constant. A new bandwidth trace was generated in which the link bandwidth varies from 20KBps and 80KBps. This trace is shown below.

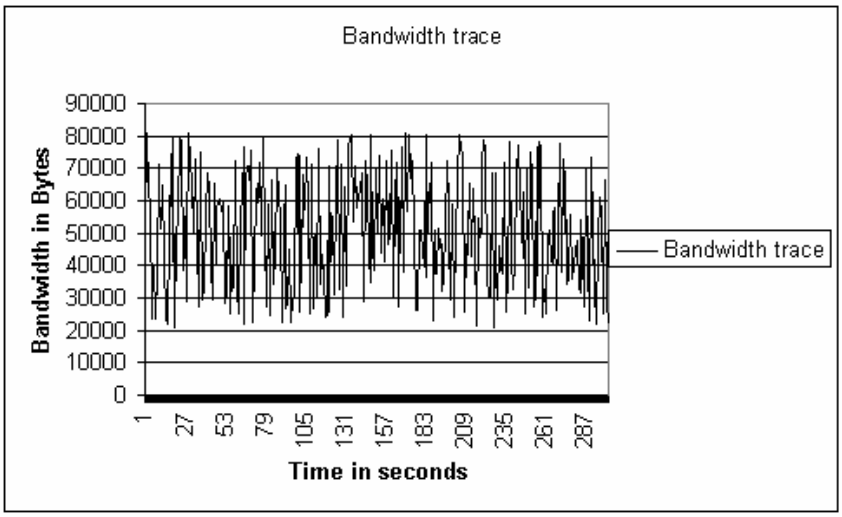

Figure 5.11: Bandwidth trace showing variable bandwidth between 20 KBps and 80KBps

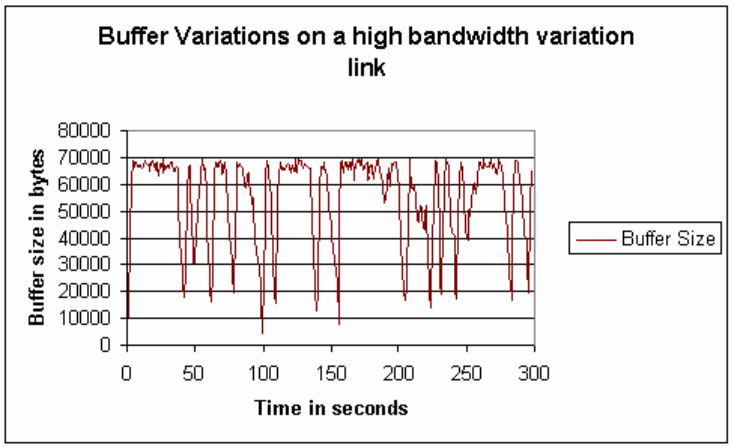

Figure 5.12: Buffer size required, transcoder and frame dropper employed, high bandwidth variation link.

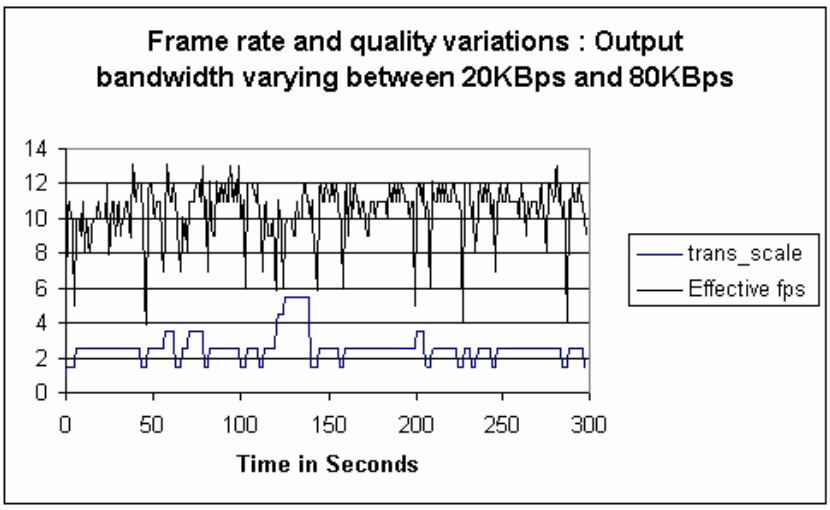

Figure 5.13: Frame rate and quality variations, transcoder and dropper employed, highly variable bandwidth link

Even though the frame rate observed is not as high as that using only transcoding, quality is maintained at approximately 2.5 fps. Compared to transcoding in which a surge in data will mean dropping of all extra frames leading to errors in the stream , dropping of B frames helps maintain smooth video.

5.2.2 Bandwidth Manager at The Client

When the manager is at the client, it can monitor the number of frames per second actually received over the network. Based on that it can send a feedback to the transcoder located at the gateway to increase or decrease the bit rate of the stream. A reliable connection was used to send the feedback over the network.

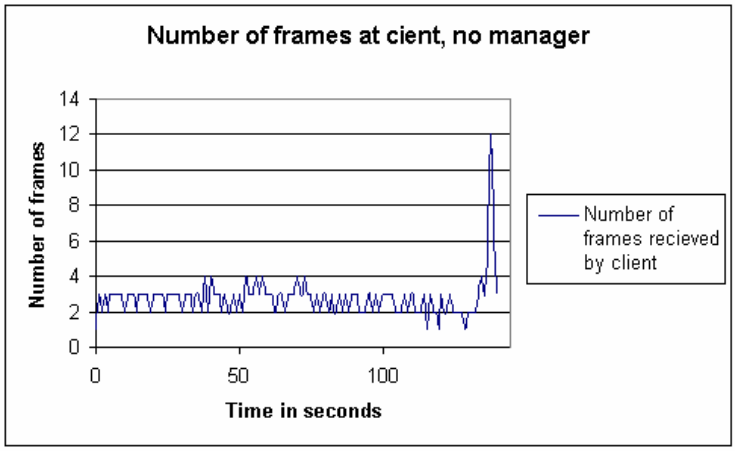

Figure 5.14: Number of frames received by client without a bandwidth manager

The above graph shows the frames per second at the client, without a manager. The link used in this case has a bandwidth of 60000 bytes per second. The frames per second is quite low, around the 3-4 fps mark on an average. The following graph shows the frames per second at the client, when a manager is employed. The transcoding rates are also specified. The manager tries to maintain a frame rate of 10-12 frames per second. A maximum transcoding rate of 7 is required.

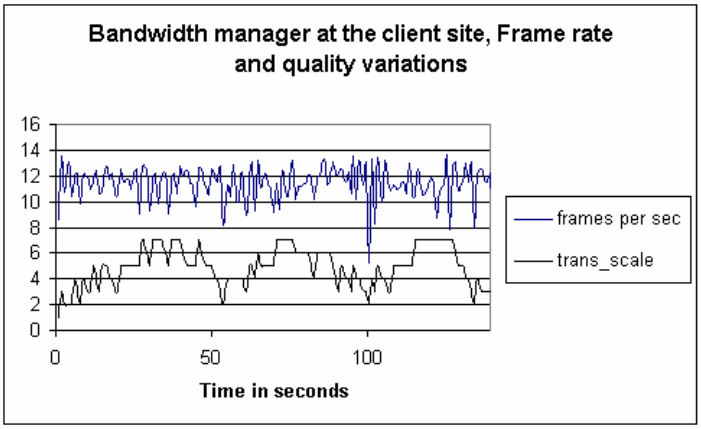

Figure 5.15: Frame rate and quality variations with the manager at the client site

5.3 Differences Between Having The Manager at The Client And at The Gateway.

When the bandwidth manager is at the gateway, the goal is to match the output bandwidth without a loss of data. With the help of the transcoder, the bandwidth manager can decrease the quality of the stream by decreasing the bit rate. it can also alter the frame rate, maintaining the quality of the pictures. However, it has no information as to what the client is actually receiving or what it needs.

If the bandwidth manager is at the client, it knows how many frames it is receiving per second. If the requirement is to increase the frame rate, it can request the transcoder to decrease the bit rate. It can maintain the quality at the desired frame rate. So there is more control, when the manager is at the client. But, there has to be a manager for each client. Also, the feedback has to go through the network to the transcoder. There is a delay involved. Hence there is a lag in the response. Setting up the feedback channel is also expensive.

Thus, when the bandwidth manager is at the gateway between two networks, it can prevent a buffer overflow. But it is not aware of the needs of the client, a low bit rate or a low frame rate. If the bandwidth manager is at the client, it can adapt itself to a required quality and frame rate.

CHAPTER 6

CONCLUSION

This chapter summarizes the contributions of the thesis and suggests some future work possible on improving this technology.

6.1 Contributions

The MPEG-2 transcoder dynamically reduces the bit rate of an MPEG-2 stream. For a high quality stream, bit rate can be reduced by as much as a factor of 10. the bit rate reduction is lesser if the original quality is lower. The frame processing times are much lesser when compared to completely re-encoding the stream. The transcoding is done in the DCT domain itself. When run on Solaris systems, the transcoder could generate as much as 15 frames per second. This speed also depends on the spatial size of each frame. The motion vectors are re-used in the output stream, but those predications may not be exactly valid as the values of DCT coefficients have now changed. This causes a drift error in predicted frames. This has been done to avoid the processing complexities involved in re-computing the motion vectors. The transcoder was used to design an adaptive bandwidth management system. The transcoder was used to manage the buffer

at the proxy as well as to maintain a steady frame rate at the receiver. When the feedback was sent from the client, it had an accurate control of the frames actually received. At the proxy, the bandwidth manager prevents buffer overflow.

6.2 Future Work

Speed optimizations can be done on the transcoder to improve frame processing speeds. The transcoder needs to be tested on different machines and also with dedicated MPEG-2 hardware. This can pump up the processing speed. Also architectures to reduce the drift error, with minimum additional overhead can be designed and incorporated to improve the quality of the transcoded stream. A method could be devised to dynamically alter the picture sizes as well, considering that palm devices and lap tops cannot handle huge images.

REFERENCES

1. Jeongnam Youn, M.T.Sun, "Motion vector refinement for high-performance transcoding", IEEE transactions on multimedia, Vol.1, March 1999, pp.30-35
2. P.N.Tudor, O.H. Werner, "Real Time transcoding of MPEG-2 Video Bit streams", IBC 97, IEE Conference Publication
3. Suresh Gopalakrishnan, Daniel R, Maximillan O., "Real Time MPEG Syatem Stream transcoder for Heterogeneous Networks", C&C Research labs, NEC USA.
4. N.Bjork, Charilaous C, "Video transcoding for UMA access", Ericcson radio system
5. E.Amir, S. McCanne, H.Zhang, "An application level video gateway", ACM Multimedia 1995 – Electronic Proceedings
6. Assuncao, Ghanbari , "Post processing of MPEG2 Coded Video for Transmission at lower bit rates" , Proceedings IEEE, ICASSP'96 , vol4 , May 1996.
7. ISO/IEC JTC1/SC29/WG11 Coding of moving pictures and audio , "MPEG-2 Generic coding of moving pictures and associated audio information".
8. Wu-chi Feng, B. Krishnaswami, A. Prabhudev, "Proactive Buffer Management for the Delivery of Stored Video Across Best-Effort Networks", ACM Multimedia Conference 1998, Bristol, UK, September 1998.
9. University of California , Berkeley, MPEG-2 Source code and Documentations.