Evaluating and Improving the Performance of Video Content Distribution in Lossy Networks

TIMOTHY RICHARD PORTER

PhilosophiæDoctor (PhD), DPhil,..

ASTON UNIVERSITY

November 2010

Aston University

Evaluating and Improving the Performance of Video Content Distribution in Lossy Networks

TIMOTHY RICHARD PORTER
PhilosophiæDoctor (PhD), DPhil,..
November 2010

The contributions in this research are split in to three distinct, but related, areas. The focus of the work is based on improving the efficiency of video content distribution in the networks that are liable to packet loss, such as the Internet.

Initially, the benefits and limitations of content distribution using Forward Error Correction (FEC) in conjunction with the Transmission Control Protocol (TCP) is presented. Since added FEC can be used to reduce the number of retransmissions, the requirement for TCP to deal with any losses is greatly reduced. When real-time applications are needed, delay must be kept to a minimum, and retransmissions not desirable. A balance, therefore, between additional bandwidth and delays due to retransmissions must be struck.

This is followed by the proposal of a hybrid transport, specifically for H.264 encoded video, as a compromise between the delay-prone TCP and the loss-prone UDP. It is argued that the playback quality at the receiver often need not be 100% perfect, providing a certain level is assured. Reliable TCP is used to transmit and guarantee delivery of the most important packets. The delay associated with the proposal is measured, and the potential for use as an alternative to the conventional methods of transporting video by either TCP or UDP alone is demonstrated.

Finally, a new objective measurement is investigated for assessing the playback quality of video transported using TCP. A new metric is defined to characterise the quality of playback in terms of its continuity. Using packet traces generated from real TCP connections in a lossy environment, simulating the playback of a video is possible, whilst monitoring buffer behaviour to calculate pause intensity values. Subjective tests are conducted to verify the effectiveness of the metric introduced and show that the results of objective and subjective scores made are closely correlated.

Keywords: Reed Solomon, Pause Intensity, H.264, Data Partitioning

Contents

List of Tables

List of Figures

List of Abbreviations

ACK	Acknowledgement
ADSL	Asymmetric Digital Subscriber Line
ARP	Address Resolution Protocol
ARQ	Automatic Repeat Request
ASIC	Application Specific Integrated Circuit
B frame	Bi-direction Predicted Frame
BER	Bit Error Rate
BSC	Binary Symmetric Channel
CIDR	Classless Inter-Domain Routing
CIF	Common Intermediate Format
CSV	Comma Separated Value
dB	Decibel
DCR	Degradation Category Rating
DCT	Discrete Cosine Transform
DOG	Digital On-screen Graphic
DVB	Digital Video Broadcast
FEC	Forward Error Correction
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GOB	Group Of Blocks
GOP	Group Of Pictures

LIST OF ABBREVIATIONS

HDL	Hardware Description Language
HTTP	Hypertext Transfer Protocol
I frame	Instantaneous Decoder Refresh Frame
I/O	Input/Output
IC	Integrated Circuit
ICMP	Internet Control Messaging Protocol
IDR	Instantaneous Decoder Refresh
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ITU	International Telecommunications Union
JVT	Joint Video Team
LAN	Local Area Network
MAC	Medium Access Control
Mbps	Megabits Per Second
MDS	Maximum Separable Code
MOS	Mean Opinion Score
MPEG	Motion Picture Experts Group
MSE	Mean Squared Error
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
NetEM	Network Emulator
NIC	Network Interface Controller
NRI	Network Reference Indicator
OS	Operating System
OSI	Open Systems Interconnect

LIST OF ABBREVIATIONS

OTS	Off-The-Shelf
P frame	Predicted Frame
PCI	Peripheral Component Interconnect
PHY	Physical Interface
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
QoS	Quality Of Service
RDT	Real Data Transport
RFC	Request For Comments (document)
RLE	Run Length Enconding
RS	Reed Solomon
RTP	Realtime Transport Protocol
RTSP	Realtime Streaming Protocol
RTT	Round Trip Time
SNMP	Simple Network Messengering Protocol
TC	Traffic Control
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
VLC	Video LAN Client
VoIP	Voice Over Internet Protocol
XOR	Exclusive Or

Chapter 1

Introduction

Not so long ago, the capability of packet-based video streaming was limited to use by large organisations, who used point-to-point video-streaming for conferencing. This was primarily due to the initial cost of the hardware and the limited bandwidths available. At the time, *Integrated Services Digital Network* (ISDN) lines were not only expensive, but extremely slow by today's standards. Available speeds were in multiples of 64kb/s, which was the minimum required for a low resolution video conference. On occasion, this primitive technology is still used today for correspondent reports and live news broadcast where a satellite uplink or internet connection is otherwise unavailable.

In recent years the amount of streamed video content available on the Internet has increased dramatically, primarily due to the availability of low-cost broadband in the domestic market. Youtube [12] and other such video streaming websites have made a huge impact on the way video content is delivered and viewed, and has made it accessible for home users.

With the illegal, and uncontrollable, distribution of copyright video content increasing, broadcasters have realised they need to provide content in an equally appealing alternative to protect their content and revenues. BBC iPlayer [13] is one such example of video content distribution, but is mostly restricted to viewing on a computer. As the on-demand revolution becomes more apparent, broadcasters will have to invest more into content distribution using the Internet, in order to satisfy the demands of the public, and to maintain a market share in online video delivery.

In this research, several potential issues are examined that the transition to wide-scale online video distribution could pose, and some beneficial enhancements are proposed to the protocols and technologies currently available.

1.1 Challenges and Motivation

Efficient transmission of video over the Internet poses several challenges. Firstly that the video must be compressed in order to produce a usable data rate. This is generally a lossy process, so the amount of compression must be carefully balanced to achieve a high quality image with reasonable bandwidth requirements. This, however, introduces additional problems if more advanced features of compression are implemented, since the video stream becomes more sensitive to packet losses.

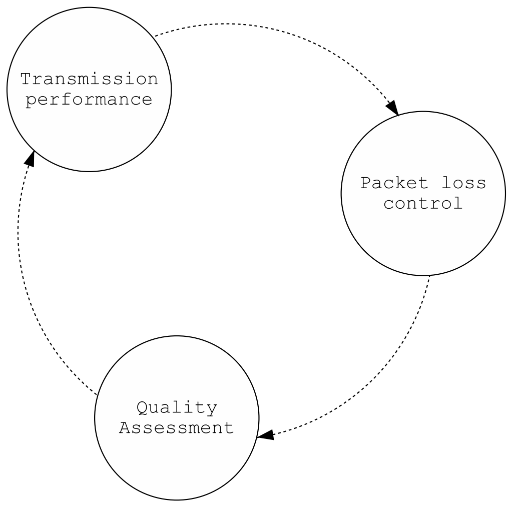


Figure 1.1: The relationship between aspects of this research

There are two main transport protocols available for the Internet, unfortunately neither are perfect for the purposes of transporting video content, especially since the Internet itself is classed as a best-effort network. This means that data is not guaranteed to arrive at the destination, and so the requirement for loss recovery or error detection protocols become apparent. The *User Datagram Protocol* (UDP) provides a method of transport for when loss is preferable over delay, since the protocol is unable to control any packet losses. As such, UDP is particularly suited for realtime applications. The *Transmission Control Protocol* (TCP) provides guaranteed end-to-end transmission due to the ability of loss recovery using re-retransmission techniques, however can introduce significant delay and cause a lowered throughput.

As such, videos are either subject to picture degradation, or continuous pausing when channel conditions are poor. Both of these disadvantages are looked at throughout this research to improve performance on both the network, and the video playback itself.

The relationship between all areas of research conducted throughout this Thesis is shown in Fig. 1.1. There are three key areas of study, transmission performance, packet loss control and quality assessment, which will be the main approaches. These are all outlined in detail in the following sections.

1.2 Aims and Objectives

The aim of this research is to present a coherent study of the transport of video content over lossy networks, and demonstrate several new methodologies for evaluating and improving the performance of transmission.

Efforts are concentrated on packet loss because of the sensitive nature of video streaming and the recent increase in video streaming seen on the Internet, which is typically a congested medium.

Both TCP and UDP will be used and improved upon, with effort to reducing the effect of packet loss. In addition, it is aimed to find novelty in areas of quality assessment, especially for video playback which is subject to continual re-buffering.

1.3 Novelty

The work presented in this Thesis is different from other research in the following ways:

- The use of loss recovery in combination with packet retransmission, demonstrated in Chapter 3, is specifically applied to the case of video content distribution. The results show a significant decrease in the number of re-transmissions required, leading to clear benefits in the real world example demonstrated.

- The new method of transport in Chapter 4 highlights the advantages of using a hybrid TCP/UDP approach, specifically for video content distribution. Since the most critical parts of the video stream can be protected in full against loss, additional video coding features can be implemented. These are shown not only to improve playback quality, but reduce the overall video data rate.

- The novel metric defined in Chapter 5 allows assessment of video quality based on playback continuity since, for the case of loss-protected transport protocols, picture degradation is not seen. Existing quality assessment methods therefore do not apply. This new objective measurement is further validated by correlation with extensive subjective results.

1.4 Structure of Thesis

This Thesis is split into a number of chapters. Chapter 2 provides preliminary background to the subject area, which is relevant to the work throughout this research. It begins with an in-depth review in Section 2.1 of the infrastructure of IP-based networks, detailing on the layers of the OSI model, for which the IP stack is based. The two main transport protocols, the *User Datagram Protocol* (UDP) and the *Transmission Control Protocol* (TCP) are discussed in detail, including flow control, congestion, and the problems associated with packet loss. The primary effect that packet loss has on both schemes is data loss for the case of UDP, and a reduction in throughput for TCP. Both these points are referred to heavily throughout the Thesis.

This is followed in Section 2.2 by video compression techniques, to outline the requirement for video coding in order to reduce the data rate, which is important for storage and streaming applications. The compression techniques discussed are all part of the original H.261 coding standard, many of which are still adopted today. Intra-frame and Inter-frame coding is discussed, along with frame prediction, chrominance subsampling, quantisation, *Run Length Encoding* (RLE) and Huffman encoding. This is followed by an overview of additional features provided by the H.264 standard, including the *Network Abstraction Layer* (NAL) and data partitioning, both of which are exploited in the contributory research. In addition, the compression efficiency of H.264 is compared to that of MPEG2 to demonstrate the primary benefit to the new standard.

In Section 2.3, the previous two sections are brought together to present background on video streaming over IP-based networks. In this, both transport protocols are discussed with respect to video transmission, and the limitations of each are made clear. Quality degradation in terms of picture corruption and persistent playback pausing are introduced, along with the methods to measure such imperfections objectively. The objective measurements are based on the *Peak Signal-to-Noise Ratio* (PSNR) and a new metric, pause intensity, which will be defined. In addition, the *Mean Opinion Score* (MOS) is introduced as a further preamble to the work in Chapter 5 which uses a MOS to validate the novel pause intensity metric.

Section 2.4 provides background on channel coding techniques which are used to proactively protect against data loss. *Forward Error Correction* (FEC) methods such as parity check codes, linear block codes, and convolutional codes are all discussed, including the theory of Reed Solomon coding schemes, which are used throughout Chapters 3 and 4.

Chapter 3 will look at the benefits and limitations of content distribution using TCP with FEC. A practical approach to combining TCP with FEC is presented, taking into account resulting overheads and therefore throughputs at different rates of packet loss. The purpose of the research is to emphasise the importance when balancing the two technologies in a practical situation. To ensure the efficient combination of both adding redundancy and requesting retransmissions, adaptive coding is the primary consideration.

Furthermore, we propose the potential for future work in this area by investigating a hardware-based extension to this particular area of research, and present an in-depth methodology as the basis for continuing development.

In Chapter 4 a hybrid TCP/UDP transport protocol is proposed, specifically for H.264 encoded video, as a compromise between the delay-prone TCP and the loss-prone UDP. It is argued that the playback quality at the receiver often need not be 100% perfect, providing a certain level is assured. Reliable TCP is used to transmit and guarantee delivery of the most important packets. This allows use of additional features in the H.264 standard which simultaneously provide an enhanced playback quality, in addition to a reduction in throughput. The results are compared with other methods of protection such as FEC.

In Chapter 5, a new objective measurement technique is investigated for assessing video playback quality, for the services delivered using TCP as a transport layer protocol. The new metric is defined as *pause intensity*, which characterises the quality of playback in terms of its continuity, since in the TCP environment, data packets are protected from losses but not from delays. Using packet traces generated from a real TCP connection, in a lossy environment, the playback of a video is simulated and the buffer behaviours monitored, in order to calculate the pause intensity values. In addition, subjective tests are run to verify the effectiveness of the metric introduced and show that the results of pause intensity and the subjective scores made over the same real video clips are closely correlated.

Finally, conclusions to the work are given in Chapter 6, where a summary of the research is presented. The personal contributions are clearly outlined, and the potential for future work in this area is discussed.

Chapter 2

Video Content Distribution

In this chapter, background information which is relevant to the research conducted throughout this Thesis will be presented. Network infrastructures, video compression, video streaming and channel coding will be the primary concerns, and are split into separate sections. Chapters 3, 4 and 5 will draw on the knowledge presented in these areas.

2.1 Network Infrastructure

This section discusses related networking background, which is relevant in all parts of the research presented. The *Open Systems Interconnect* (OSI) model is introduced along with its associated layers, in particular the Transport and Network layers, which provide end-to-end link control and inter-network packet routing, respectively. Protocols such as TCP and UDP will be outlined in detail, in addition to the side effects on transmissions, which congestion and packet loss may have. Particular side effects include throughput, delay and the reordering of packets, which are often found on best-effort networks such as the Internet.

The OSI model is described in detail since much of the research conducted requires knowledge of each layer as a prerequisite.

2.1.1 The OSI Model

The OSI model [14] is a standard provided by the Open Systems Interconnection which was developed by the International Organization for Standardisation. It allows for the separation of constituent parts of a communications network into a series of layers, each tasked with a particular function. The relationship between layers is made up of interfaces, which allow neighbouring communication, however, each layer is independent of the other, which provides a high level of abstraction between protocols.

The model is most commonly adapted by the *Transmission Control Protocol/Internet Protocol* (TCP/IP) stack, which provides packet-based communication over a connectionless network of hosts. The model itself is used by protocols as a generally accepted outline, but is not a requirement of the specification. The complete OSI model used for the TCP/IP stack is shown in Fig. 2.1.

A network is said to be connectionless if communication is performed between two end-points without prior arrangement. There is no handshake nor session established before transmission occurs, and flow control does not exist. The order of data during transmission, therefore, is not necessarily preserved.

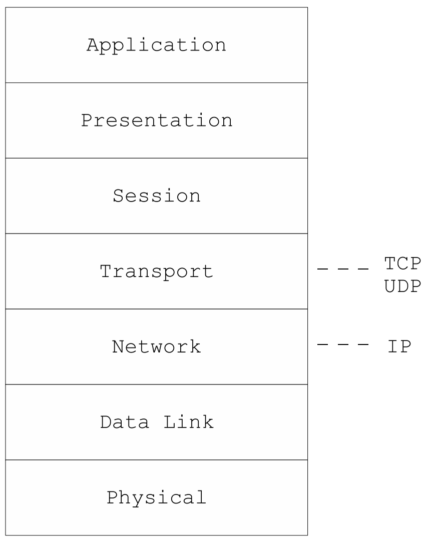


Figure 2.1: The OSI and TCP/IP layered model

The **Application layer** is the uppermost layer in the model and provides an interface for User-space applications such as web and email clients to invoke protocols such as the *Hyper Text Transfer Protocol* (HTTP) and *Internet Message Access Protocol* (IMAP). It has the highest level of abstraction, providing little insight into the network below.

The **Presentation layer** deals with translation of differing formats. For example *Ameri-*

can Standard Code for Information Interchange (ASCII) and Unicode representation of plain text. It provides the application layer with consistent content, abstracting the method that was used to encode it.

The **Session layer** provides the ability to maintain and identify the state of a connection between two hosts, even if no data is being transmitted, or if one of the hosts becomes momentarily unreachable.

The **Transport layer** provides a wrapper for data reliability; its primary purpose is to ensure data is framed and transmitted successfully to the destination host. The *route* of transmission is unknown, since this is dealt with at lower layers, so the two ends of the link assume to communicate directly. Protocols such as the *Transmission Control Protocol* (TCP) provides end-to-end connection control, allowing retransmission of lost packets and congestion control. The *User Datagram Protocol* (UDP) does not have these properties, however detection of corrupt packets is possible using a header checksum. Both of these transport protocols are discussed in detail in following sections.

The **Network layer** is responsible for routing the packeted data across the network, primarily by determining whether or not the destination host is local. This is achieved by analysing the packet's destination IP address to determine the sub-network, or *subnet*. If the host address is within the local subnet, it is simply passed on for delivery to the locally determined *Media Access Control* (MAC) address, otherwise it is destined for the border gateway of the *Local Area Network* (LAN), and routing to the appropriate network. MAC addresses are determined using the *Address Resolution Protocol* (ARP). This layer is the highest level of knowledge in a router.

The **Datalink layer** provides an interface between the network layer and the MAC layer, such as with Ethernet, before on-wire transmission at the physical layer. The data is framed into *bits*, suitable for transmission on the particular physical medium. Start and stop bit sequences are used, and error correction and flow control is applied. The MAC layer can deal with the collision of data at the physical layer, but this is typically left to hardware-based carrier sensing on modern switches.

The **Physical layer** is the lowermost end of the model. It describes the standard of transmission of data at voltage-level (if copper is used) using the bits provided by the datalink layer as modulation.

2.1.2 Internet Protocol

The *Internet Protocol* (IP) allows for the routing of datagrams (information within a packet) across many interconnected Wide Area Networks (WANs), and is what makes up the Internet. In addition, IP provides addressing (namely IP addresses) which identify individual hosts on a network in a logical manner. Each host is assigned a distinct address from all other hosts on a LAN, and are usually (for IP version 4) in the form of a dotted-decimal (or dotted-quad). For example `aaa.bbb.ccc.ddd` where *a* to *d* are bounded integers. Each quad must not exceed the value 255 due to representation by a single byte in the packet. The permissible values are defined by the *Internet Engineering Task Force* (IETF) in [2].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.2: The header of an IP packet [2]

The data within an IP packet is appended to a header, which provides information about the source and destination of the data, in addition to the length, checksum and routing parameters. This is shown in Fig. 2.2. The most significant fields from this header are the source and destination fields, which are IP addresses.

Each host is assigned an IP address and a subnet *mask*, which is used to divide the address into a network prefix (or sub network) and host identification. The subnet mask is defined in the same format, and with the same constraints, as the IP address itself, but each quad specifies how many bits within the address should be masked off for network identification. For example an IP address of `aaa.bbb.ccc.ddd` with a subnet mask of `255.255.255.0` denotes `aaa.bbb.ccc` is the network address, and `ddd` is the host address. Split-byte masks can also be specified, for example `255.255.192.0`, expressed in binary form as `11111111.11111111.11000000.00000000` provides a mask of the first two bytes and the two most significant bits of the third byte. The host addresses are therefore limited to the bounds of that mask, when an address is assigned.

The addition of a subnet mask to an IP address is often conveniently represented using the *Classless Inter-Domain Routing* (CIDR) notation, in the form `aaa.bbb.ccc.ddd/n`, where `n` is the number of bits which have been masked for the network address.

2.1.3 User Datagram Protocol

The *User Datagram Protocol* (UDP) is, on the most part, an IP packet with additional headers to allow transportation of the data. The structure of the header is shown in Fig. 2.3.

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...
```

Figure 2.3: The structure of a UDP packet [3]

The port numbers provide the source and destination end points during transport. When the packet arrives at the destination, the packet payload data is passed to the process listening on the specified port. The source port is an optional requirement, however the destination port is mandatory. UDP does not offer flow nor error control; indeed the header checksum is optional. The length field specifies how many bytes are expected in the header and payload combined. All fields in a UDP header are 2 bytes (16 bits).

UDP is generally used when transmitting infrequent messages over a long period, since there is no connection tracking [3]. For example, in combination with the *Simple Network Messengering Protocol* (SNMP), which is used primarily for logging and realtime monitoring of hosts on a network. In addition, UDP is particularly useful for realtime video transport due to the independent transmission of each packet, and is often used in combination with the *Realtime Transport Protocol* (RTP). This will be discussed in detail in a later section.

Applications which use UDP must expect some degree of data loss and ordering displacement since the transport is considered unreliable. If a measure of error control is required with UDP, for example, when transferring files using the *Trivial File Transfer Protocol* (TFTP), then this must be implemented in User-space at the Application layer.

2.1.4 Transmission Control Protocol

The *Transmission Control Protocol* (TCP) is the most commonly used transport protocol in combination with IP. TCP was originally specified to run separately, IP being used for essentially what UDP is now, however it was redeveloped at an early stage to run together [4]. As such, TCP is commonly referred to as TCP/IP.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgement Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.4: The header of a TCP packet [4]

As a transport protocol, TCP provides several important features which have become the backbone of flow and error control on the Internet. Unlike UDP, which was discussed earlier, TCP has mechanisms in place to control the packet loss and rate of transmission using a feedback channel.

With just the TCP headers and the payload, the data is collectively known as a TCP segment. When the segment is passed to the network layer, IP headers are added and the packet becomes ready for passing on to the data link layer.

The header of a TCP packet is shown in Fig. 2.4. Notable fields within the header are the *Sequence Number*, *Acknowledgement Number* and *Window Size*. The sequence number allows each packet to be tracked, providing clear indication to the receiver should one become lost in transmission. The acknowledgement number allows the receiver to indicate which packet it is expecting next, and provides the transmitter with confirmation of successful delivery up to that point. The window size indicates how much data, in bytes, the receiver is willing to accept in one segment. This provides flow control, which Section 2.1.4 will detail.

Automatic Repeat Request

Packet loss is controlled by the implementation of a *positive acknowledgement* (ACK) for each packet received by the destination. With the reception of an ACK, signifying a delivery success, the sender continues to send the next packet. If the ACK is not received, or the previously sequenced ACK is received again, then the data is retransmitted. This functionality is called the *Automatic Repeat Request* (ARQ), which is illustrated in Fig. 2.5.

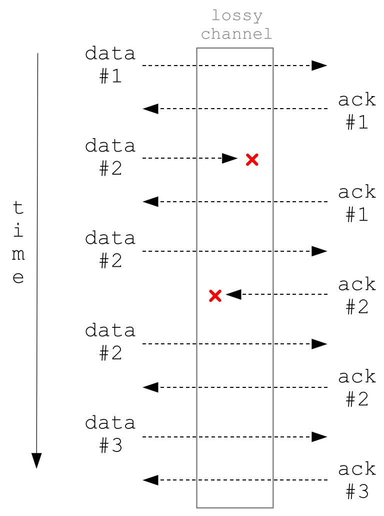


Figure 2.5: The functionality of ARQ

In this example, data packet #1 is received and acknowledged without issue. This is the normal functionality of TCP. Data packet #2, however, is lost during transmission which causes the receiver to once again acknowledge data packet #1, after a specified timeout. Receiving the first ACK again, the sender will retransmit data packet #2. Additionally, if the ACK is not received in a timely manner, the sender will retransmit.

If the delay between sender and receiver is T, then the time between transmission of the data and reception of an ACK is at least $2T$, and called the *Round Trip Time* (RTT). The RTT is typically measured in milliseconds.

Flow Control

Since the stop-and-wait behaviour of positive acknowledgement requires waiting twice the transmission delay before a packet can be deemed successfully received, the use of this protocol yields poor utilisation of bandwidth, even under good conditions. As such, the sliding window protocol is used to allow the sender to transmit multiple packets in succession, even if an ACK from a previous packet is yet to be received. The advantage to this methodology is illustrated in Fig. 2.6.

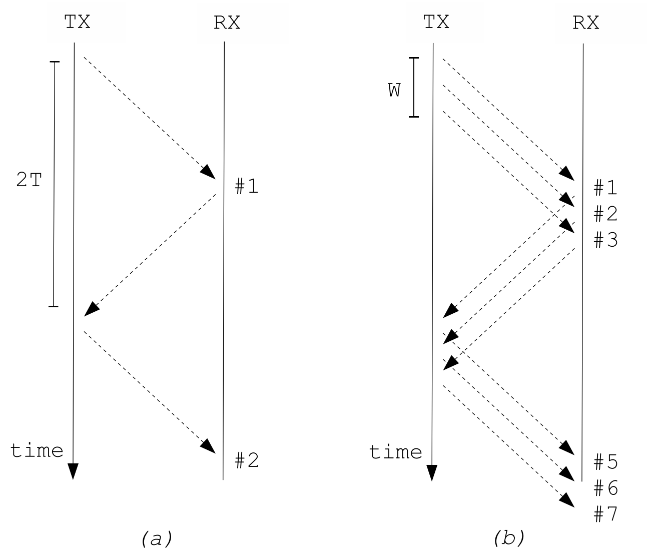


Figure 2.6: Stop-and-wait *(a)* vs. a sliding window protocol *(b)*

The sliding window approach (Fig. 2.6*(b)*) provides more efficient transport of packets than the stop-and-wait algorithm (Fig. 2.6*(a)*). The number of unacknowledged packet transmissions at any one time is called the window size, W, which is a discrete positive number. With each packet ACK, the receiver specifies how much data it is willing to buffer. Accordingly, the transmitter will adjust the window size so as not to exceed the limit imposed. The *Maximum Segment Size* (MSS) is the amount of data TCP is willing to transmit in one packet. This is usually close to the *Maximum Transmission Unit* (MTU) of IP, but must allow for additional headers to avoid fragmentation by the Network Layer. Packet loss occurs either due to noise on the transmission line, or packet-discard at a congested router [5].

There are several issues which become apparent with retransmissions when sending multiple packets at once. Most predominantly, what should occur if a packet is unacknowledged? The two most common protocols are *go back n* and *selective repeat*. Fig. 2.7 illustrates the functionality of both protocols.

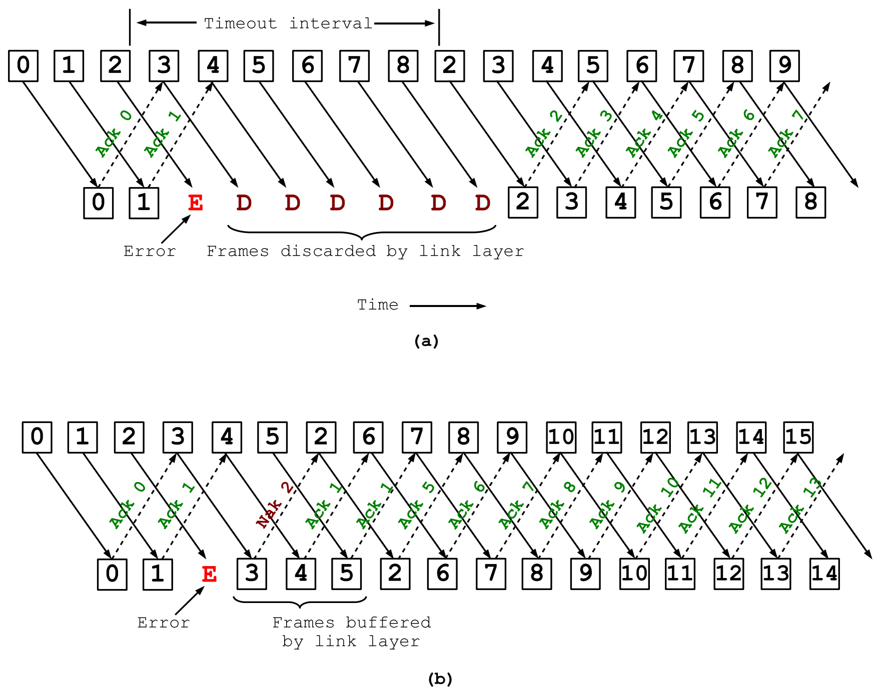


Figure 2.7: Go back n *(a)*, and selective repeat *(b)* protocols [5]

In the case of *go back n* (Fig. 2.7*(a)*), packet 2 is seen to be lost during transit and therefore not acknowledged by the receiver. Since a sliding window protocol is being implemented, the transmitter continues to send data until a timeout occurs for the original ACK. If the number of packets transmitted between the last received ACK and the timeout is *n*, then the transmitter goes back *n* packets and resends from position 2. At the receiver, the data link layer drops all packets received since the initial loss and awaits the next expected packet in the sequence. The transmitter therefore needs to buffer any unacknowledged packets in case a retransmission is required.

For *selective repeat* (Fig. 2.7*(b)*), lost packets remain unacknowledged, however the packets which are received ahead of sequence are buffered by the receiver. When a timeout occurs for the unreceived ACK, the transmitter performs a retransmission of the lost data. The receiver sends an ACK for every packet received, even if out of order, but the sequence number within the ACK will remain as the last successfully received *in-sequence* packet.

A notable improvement to TCP flow control came about when the Nagle algorithm [15] was developed. This attempted to ensure that the TCP segment size was close to the MSS to avoid packets with small payloads. Since the protocol header sizes are fixed, sending a single byte payload with a 40-byte header (20 bytes for TCP, 20 bytes for IP) is an extremely inefficient use of bandwidth.

Congestion Control

When a packet is lost during transmission, it is assumed by TCP that the loss is due to congestion on the network between source and destination. As a reactive measure, to prevent congestion collapse [15], the window size is typically halved at the transmitter, reducing the number of packets which are sent before acknowledgement. In addition, if a router on the transmission path wishes to, it may send a source quench packet to the transmitter, which will have the same effect of window size reduction. The source quench packet, sometimes called a choke packet, is sent using the *Internet Control Message Protocol* (ICMP) [16].

TCP variations

There are several different versions of TCP which have been developed, mainly to increase performance under congested conditions [17] [18] [19]. However, additions to the original standard must be backwards compatible with previous versions of TCP, to maintain a heterogeneous network. As such, changes in the specifications are usually limited to the congestion control algorithm at the sender, such as the rate of reduction of the window-size when packet loss is detected.

2.2 Video Compression

Uncompressed video poses a potential problem for storage and streaming solutions alike. Since each frame in a video sequence is produced using still images, and each image of a series of pixels, the amount of data required to represent a video is substantial. The resolution of *Common Intermediate Format* (CIF) video is 352x288 pixels which, when using an RGB colour space, requires three bytes per pixel to represent the luminosity values for red, green and blue. One byte per pixel provides a 24-bit true colour image. Each CIF resolution frame therefore requires 304,128 bytes which, at a standard 24 frames per second, amounts to 58.4 *Megabits per second* (Mbps). This is clearly an unachievable data rate to transmit over a conventional network; indeed at higher resolutions such a DVD quality (720x576), or high definition (1920x1080), the data rate rises to 239Mbps and 1194Mbps, respectively. With this in mind, a number of lossy video compression standards have been produced to allow a significant decrease in the required data rate.

There are two primary organisations who have developed the majority of video coding standards, namely the *International Telecommunications Union* (ITU) and the *Motion Picture Experts Group* (MPEG). In recent years, the two organisations have combined to produce the *Joint Video Team* (JVT) and were responsible for producing the widely used MPEG2 specification, and the newer, highly efficient H.264/MPEG-4 Part 10 specification. This new standard has a reported saving of more than 50% in data rate for the same quality video, compared to the older MPEG2 standard [20]. The enhancements made in the H.264 standard, which have made this achievable, will be discussed later.

This section will provide background in the video compression techniques which have been developed over recent years. In particular, the H.264 coding standard is discussed, the most recent video coding specification, and which is used throughout the research conducted.

2.2.1 Compression Techniques

The introduction outlined the importance of compression to allow video streaming using a feasible data rate. H.261 was the first notable video compression standard, ratified in 1988 [21] by the ITU. All subsequent international video coding standards are based on the fundamental principles from this specification.

The specification was originally developed for video conferencing over *Integrated Services Digital Network* (ISDN) lines and, as a consequence, permitted data rates in multiples of 64kbps, up to a maximum of 1920kbps [6]. Resolutions of 352x288 (CIF) and 176x144 (Quarter CIF, QCIF) were supported.

A YCrCb colour space was used in H.261 due to the benefits it provides in terms of compression and human perception. Rather than having a separate channel for red, green and blue (as for the case with RGB), the luminance (brightness) and chrominance (colour) components are separated to give a YCrCb colour space, where Y is the luminance, and Cr/Cb are the red-difference and blue-difference components from the Y channel. This information is sufficient enough to reconstruct a full-colour frame, but provides the ability to compress the colour components to a greater degree because the human eye is more sensitive to changes in the Y channel.

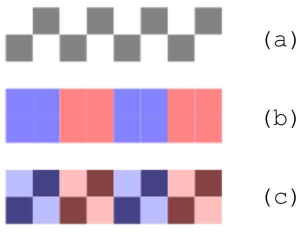


Figure 2.8: Luminance (a), Chrominance subsampling at 4:2:0 (b), YCrCb composite (c).

As such, the chrominance channels can be further subsampled, meaning a CIF resolution video may only have a QCIF colour resolution. This technique allows a reduction of the video data rate. An example of subsampling is shown in Fig. 2.8, where the luminance and chrominance information has a 4:2:0 ratio; the composite of (a) and (b) is shown in (c).

Each CIF frame is made up from 12 *Group Of Blocks* (GOBs), which contain 33 macroblocks each. The macroblocks are 16x16 pixel samples with a colour space encoding as described above. This structure is illustrated in Fig. 2.9

Intra-frame Coding

Intra-frame coding can be described as compression within the current frame only, which is termed spatial coding. These compression methods have no dependence on other frames within the video stream. The spatial compression techniques described in this section are

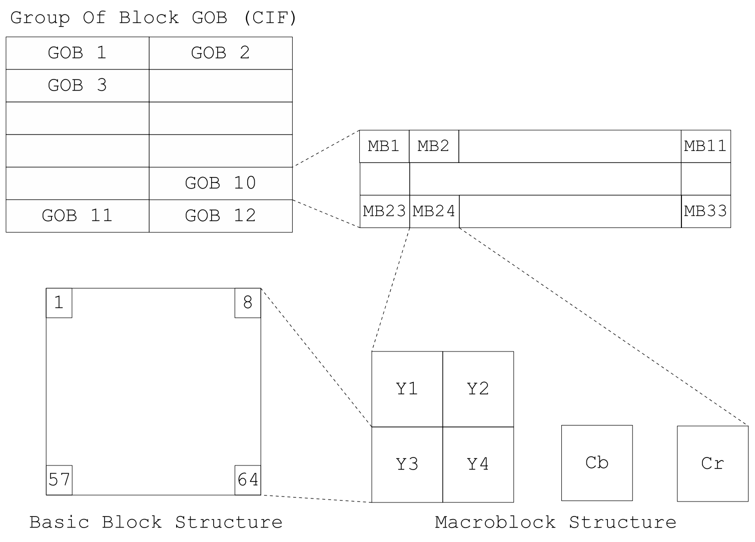


Figure 2.9: H.261 block structure [6].

done on a per-macroblock basis.

A *Discrete Cosine Transform* (DCT) is a Fourier-related operation, used to transform sections of the image from the spatial domain to the frequency domain. Each section is transformed into a $n \times n$ block, which in the case of H.261 is 8×8. The data at block position $(0,0)$ is the zero-frequency component (at DC level) for both horizontal and vertical index values, and block (n,n) contains the highest spatial frequency. Fig. 2.10 illustrates the spatial representation of high and low image frequencies in an 8×8 block, as described.

In this form, the higher frequencies can be filtered with little perceived effect to the quality of the image, once a reverse transformation is performed. This is done by quantisation of the transformed $n \times n$ block by dividing each component by a constant value and then taking only an approximate result, effectively rounding the data.

The reverse transformation is achieved using an Inverse DCT transform to return the data to the spatial domain. When the block is reconstructed, some of the higher frequency components have been removed due to the quantisation process, and so the image is simplified. This results in a reduction in the data rate of the video, allowing more efficient transmission.

A further level of compression, performed in addition to the process above, is *Run Length Encoding* (RLE). After quantisation, the byte stream is examined for long repeating sequences, particularly since many of the byte values are zero due to the removal of high-

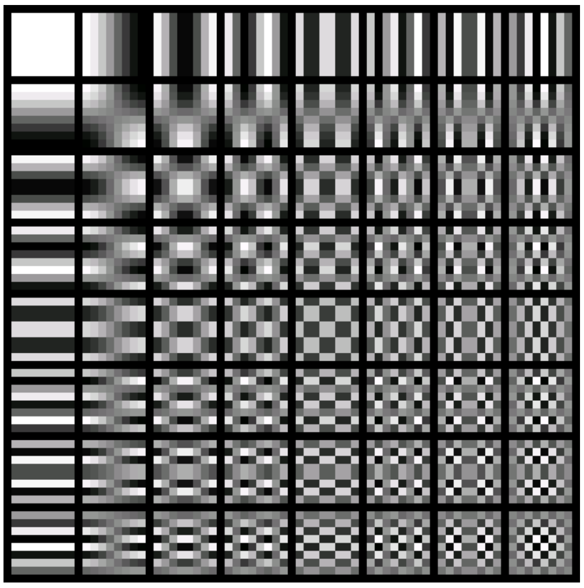


Figure 2.10: Spatial representation of low and high frequency image elements.

frequency components. The encoding process is trivially simple at this stage. If a sequence of consecutively repeating bytes are found within the stream, the sequence is replaced by a single value together with repetition factor. For example, the following byte stream

0x15 0x11 0xC9 0x45 0x00 0x00 0x00 0x00 0x00 0x00 0x71 0x1F

can be encoded to

*0x15 0x11 0xC9 0x45 0x00(**6**) 0x71 0x1F.*

The binary sequence would contain the value and repetition factor within the stream itself, but is shown as a separate integer for clarity. Use of escape values can be used to signify the start of an encoded sequence. Unlike quantisation, RLE is a lossless compression technique.

The final coding step performed in this mode is Huffman coding, which provides a lossless compression based on lookup tables. It works on the principle that the frequency of certain symbols will be more common throughout the stream than others. Hence, symbols with a greater probability of occurrence are mapped to the value, in a lookup table, composed of the fewest bits. Conversely, the least probable symbols are left to the bit lengths of longer sequences.

The quantisation process has variable compression performance based on the level of floating-point round up. A greater level of compression will lead to a great degradation in the image quality, particularly for the high frequency components. Whilst this step is classed as lossy compression, RLE and Huffman coding techniques are entirely lossless and can be applied in addition to the quantisation, providing a further reduction in the data rate of two to three times [22].

Inter-frame Coding

The previous section looked at the ways compression is achieved on a frame using only spatial coding techniques. Inter-frame coding uses data from neighbouring frames to add a further dimension of compression to the video sequence, which can reduce the overall data rate significantly [23]. This is largely due to significant levels of similar data, shared amongst temporally related macroblocks. This generates the notion of dependency amongst frames, in that the successful decode of a temporally dependant frame is subject to the neighbouring frames being available.

Consider the case in Fig. 2.11 whereby a four-frame sequence consists of a person walking from left to right, with a back drop of still scenery.

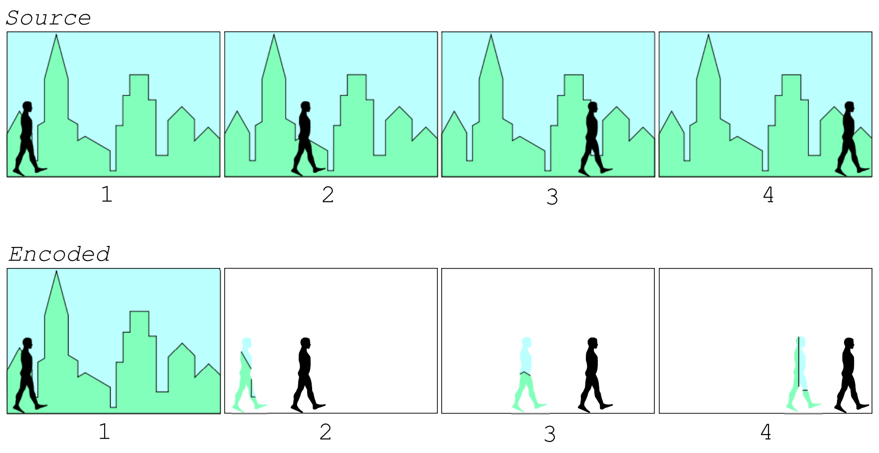


Figure 2.11: Compensated prediction using motion vectors in temporally similar frames.

Since the backdrop is not changing in frames subsequent to the first, the encoded data for this area can be reused in frames 2-4, leading to compression benefit. The moving person, which is assumed to remain constant in appearance, requires encoding from the source video for every new position, as the scene progresses. This includes the parts of the backdrop which are subsequently revealed.

There is, however, an additional technique which can be used to increase compression further, called Motion Compensation. In this case, rather than macroblocks of previous frames simply being predicted forward, a motion vector is associated with the prediction which describes a frame based on a previous image, even if the content is not positioned the same. In this case, with the example in Fig. 2.11, hardly any of the source video would be required in frames 2-4, since the predictions would be motion compensated.

The frames within the video sequence are grouped together in a predefined length to form a *Group Of Picture* (GOP). Within the GOP, there can exist both spatial and temporally dependant frames, for which there are three types. I-frames, which are intra-frame coded, and P-frames and B-frames, which are both inter-frame coded. The I, P and B are short for *Instantaneous Decoder Refresh* (IDR), *Predicted* and *Bi-directionally Predicted*, respectively.

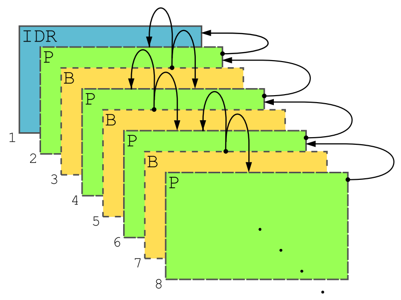


Figure 2.12: I-frames, P-frames and B-frames and their temporal dependence within a GOP.

As shown in Fig. 2.12, the GOP begins with an initial I-frame, and then a number of predicted frames thereafter. The I-frames are typically the largest type of frame in terms of encoded bits. A P-frame contains only part of the picture along with a series of motion vectors which reference data in a previous frame within the GOP.
The use of B-frames started with the MPEG1 standard, which allow for predictions based on both previous and future I- or P-frames.

Fig. 2.13 illustrates how errors in one frame can lead to frame degradation throughout much of the remaining GOP. In this case, frame 4 has sustained damage due to packet loss, which the adjacent frames, 3 and 5, depend upon. The predictions made are based on cor-

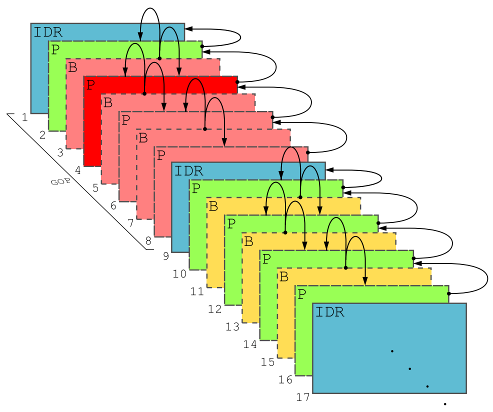


Figure 2.13: Progressive GOP corruption when a P-frame becomes damaged.

rupted data and the errors are propagated. The corruption then follows into frame 6 and beyond; the remaining frames in the GOP are all affected. The frequency of I-frames can be increased to minimise the duration of corrupt video, however this will increase the overall data rate of the video.

Fig. 2.14 shows how corruption of a B-frame does not affect other frames within the GOP since, in usual cases, predictions are not made from B-frames [8].

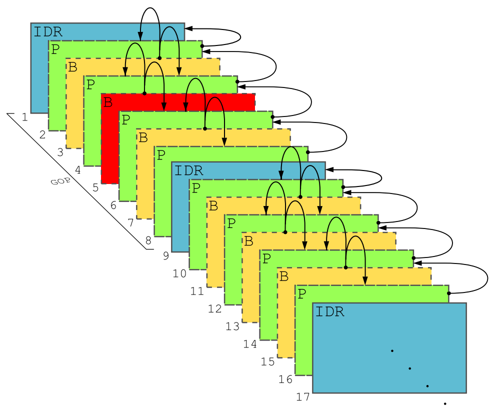


Figure 2.14: GOP corruption when a B-frame becomes damaged.

When attempting to seek through an encoded video of this type, playback may only commence from an I-frame position, or at the start of a GOP. If the size of the GOP is large, the seek resolution will be reduced.

2.2.2 H.264 enhancements

The *Joint Video Team* (JVT) is a combined effort between the ITU and MPEG. They are responsible for developing the specification for the highly efficient H.264/MPEG-4 Part 10 specification. The H.264 coding standard provides a number of enhancements over previous video *Encoder/Decoders* (codecs) such as MPEG2, and more so over H.261. The general compression techniques, however, remain the same as the earlier standards.

The primary difference in the encoding process, is the use of slices as an alternative to the GOP structure, shown earlier in Fig. 2.9. Slices can contain a variable number of macroblocks and are encoded independently of surrounding slices. As such, the spatial compression techniques, which were discussed earlier, are applied separately to multiple slices within the same frame. This has positive benefits since there becomes the potential to compress certain parts of the frame to a greater degree. An example of this can be the compression of a persistent logo throughout the video sequence, such as a *Digital On-screen Graphic* (DOG), popularly used as television channel identifiers.

In addition to a slice structure mode, the following subsections discuss the features already introduced which are particularly relevant to the research presented. In particular, the advantages of the *Network Abstraction Layer* (NAL), data partitioning and the overall compression efficiency are highlighted.

Network Abstraction Layer

The *Network Abstraction Layer* (NAL) provides an interface between the encoded video slices and the a video transport layer. This transport layer can either be optimised for disk storage or transmission using a protocol such as RTP [24]. When the latter is used, the NAL packetises the video stream data into *NAL Units* (NALU), ready for encapsulation in RTP and transmission over an IP network. The first byte of a NALU contains data to determine whether the slice that follows is used as a reference for other slices. Fig. 2.15 shows the structure of the single-byte NAL header.

The *forbidden bit* signifies a valid NALU and should always be 1. The *Network Reference Indicator* (NRI) signifies dependence on the slice if set to a non-zero value. When set to zero, the slice within the NALU is assumed to be a B-frame and so loss of this packet does not cause errors in any other frame [7]. The *type* field describes specifically the contents of the

```
+---------------+
|0|1|2|3|4|5|6|7|      F    - Forbidden bit
+-+-+-+-+-+-+-+-+      NRI  - Network Reference Indicator
|F|NRI|  Type   |      Type - NALU type
+---------------+
```

Figure 2.15: H.264 NAL header [7].

NALU, and can be used to further determine the importance of the slice if data partitioning is being implemented.

Data Partitioning

The encoded slice data, which has been assigned to a NALU, can optionally be categorised into one of three importance levels, based on the type of slice data it contains. This technique is called data partitioning, where each partition type is named either A, B or C. Partition A contains the most important data such as headers, macroblock types, motion vectors and quantisation parameters [23]. Partitions B and C contain residual information for the intra-coded and inter-coded macroblocks, respectively [23]. From this, it is clear that any loss of information from partition A can cause the video to fail during the decoding process. Additionally, partitions B and C cannot be used without partition A header information.

Chapter 4 exploits the use of data partitioning and applies unequal loss protection to the resulting NALU packet stream to ensure partition A is guaranteed to arrive at the destination. The ability to guarantee partition A data allows the use of B-frames during encoding, which would otherwise be inadvisable due to their high temporal dependence on other frames, particularly partition A slices.

Peak Signal-to-Noise Ratio

If it often necessary to measure the quality of a decoded frame, in comparison to the original, to gain an objective measurement of the difference between the two. This is achievable using the *Peak Signal-to-Noise Ratio* (PSNR) and is the method used in Chapter 4 to grade the reduction in playback quality as packet losses increase.

The measurement is most easily defined using the *Mean Squared Error* (MSE), where one frame is considered a noisy approximation of the other.

The MSE is defined as

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

The PSNR is then defined as

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \end{aligned}$$

When using this measure of comparison, it is typical to consider only the luminance channel, since it contains the majority of the image detail. MAX_I is the maximum pixel value, which is typically 255. The units are expressed in *deciBels* (dB).

Compression Efficiency

To demonstrate the compression efficiency of H.264 in comparison to an older codec such as MPEG2, a test was performed using the steps illustrated in Fig. 2.16. The quality of each video is analysed to demonstrate the higher rate of performance that the newer codec possesses.

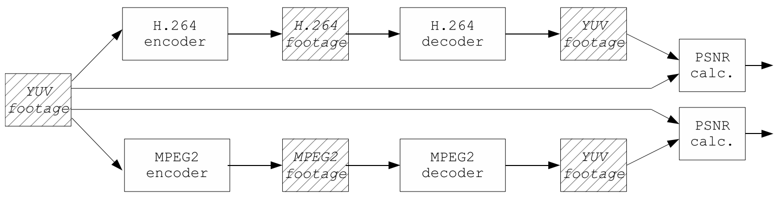


Figure 2.16: System diagram for quality comparison of two similar-content video sequences. The output is the PSNR between source and encode.

An uncompressed YCrCb video source was taken at CIF resolution and encoded to two different compression formats, MPEG2 and H.264, with identical target bit rates of 1Mbit/s. The frame structure and GOP size was the same between the two sequences, however the H.264 format utilised the specific enhancements discussed earlier. The resulting encode was then decoded back to uncompressed YCrCb for analysis using PSNR - a quality assessment measurement which will be discussed further in Section 2.3.

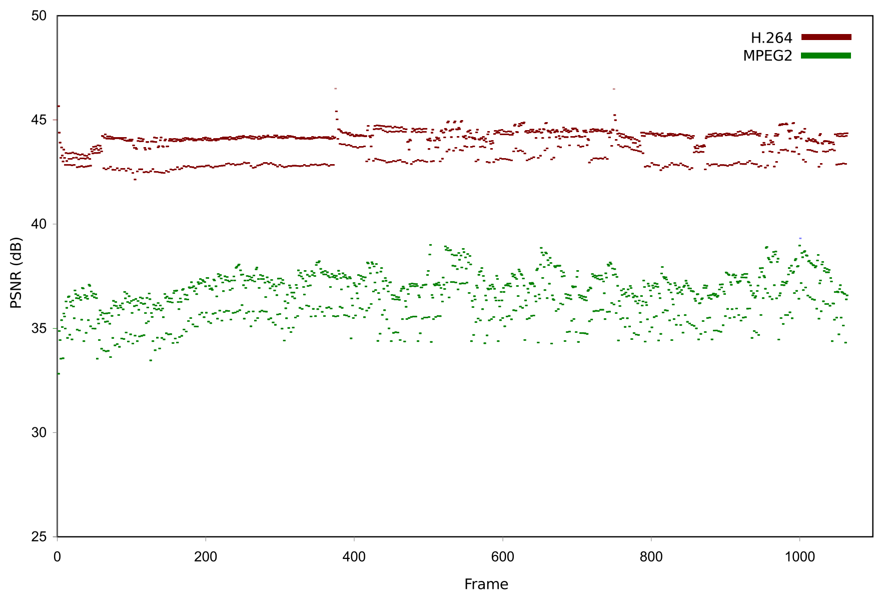


Figure 2.17: Comparison between two equal bit rate videos, one encoding using MPEG2, another with the new H.264 standard.

A higher level of PSNR denotes a greater resemblance to the source video. Fig. 2.17 clearly shows that the quality of the H.264 encode is closer to that of the source video, compared to the MPEG2 encode. Indeed, this new H.264 standard has a reported saving of more than 50% in data rate for the same quality video, compared to the older MPEG2 standard [20].

File	Lossy	PSNR	Size (bytes)	Compression
YUV source	-	-	155M	-
ZIP	no	-	116M	25%
BZIP2	no	-	94M	39%
MPEG2	yes	37.57	6.9M	96%
H.264	yes	37.53	2.6M	98%

Table 2.1: Compression rate comparisons

Table 2.1 shows experimental data for the compression efficiency of two well-known file compressors, compared with the compression efficiency of two lossy techniques, on a raw YUV file. In each lossy case, a bitrate is selected to achieve a closely matched PSNR after encoding to give comparable results. As well as the clear advantages to a lossy compression scheme, the H.264 encoder is 62% more efficient than MPEG2 for near-identical quality, which highlights the primary advantage to this modern codec.

2.3 Video Streaming

In Section 2.1, the infrastructure of IP networks was discussed in detail, which was followed in Section 2.2 by background on video compression codecs and associated techniques, in particular, the newest H.264 format. In this section compressed video is discussed, and how it is encapsulated into NAL units for streaming over an IP-based network. Several caveats associated with doing so are outlined.

The two methods of transporting video which will be detailed in this section are the UDP-based RTP format, and TCP. As will be seen, there are significantly more issues to transmitting video using UDP, fundamentally due to its lack of error control.

2.3.1 UDP-based transport

RTP is a ratified standard [3] which allows for transmission of multimedia services over IP-based networks, such as the Internet. RTP uses UDP for transport, the header of which is shown in Fig. 2.18.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.18: The header of a UDP packet [3]

The significant components of the RTP header are the sequence number, timestamp and *Synchronization Source* (SSRC) identifier. The initial fields are version, padding and optional parameters.

The sequence number provides the packetised video stream with important ordering information, a feature that UDP does not support natively. The timestamp signifies when the packet was transmitted, and the SSRC identifier provides the application layer with a unique stream identifier.

Quality Degradation

When packet loss occurs, UDP has no mechanism for recovering losses, so the integrity of the video stream will be affected. Fig. 2.19 demonstrates the effect of packet loss on a UDP-based video stream.

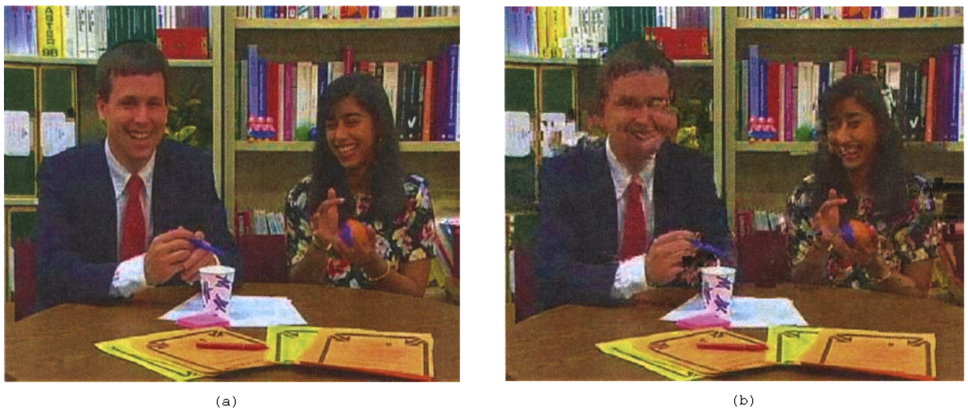


Figure 2.19: Original encoded video sequence *(a)*, subject to packet loss *(b)* [8].

The original frame is shown on the left *(a)*, and the resulting degraded frame on the right *(b)*. In this example, packets were dropped at a uniformly random rate averaging 10% [8]. This degradation in quality is obviously undesirable, and so the receiver must rely on error concealment in an attempt to mask the missing or corrupt data.

Error Concealment

Error concealment is classed as a post-processing technique because it is performed after decoding of the video data. Typically, packets are lost which cause a macroblock or slice to become corrupt. Since the corrupt data cannot be decoded, the area in question becomes blank. Fig. 2.20 illustrates the two main methods of concealing a missing macroblock, firstly through spatial concealment *(a)*, and secondly by temporal concealment *(b)*.

Spatial concealment takes the picture information from neighbouring macroblocks in an attempt to mask the missing block, which can often be highly effective in areas of low detail. Temporal concealment uses picture information from adjacent frames in the GOP to perform linear extrapolation on the missing block, which is effective when the difference between frames is minimal.

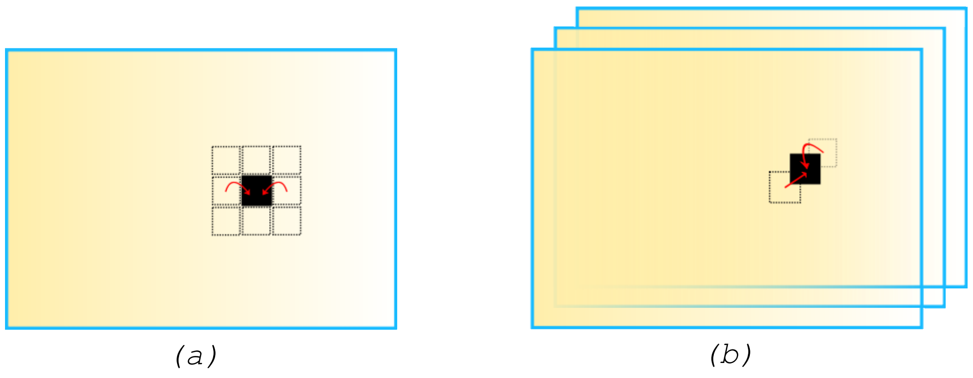


Figure 2.20: Error concealment using spatial *(a)*, and temporal *(b)* algorithms.

Should spatial or temporal concealment not be possible due to excessive data loss, the final approach is to *frame copy* the last good frame, for display until successful decoding can resume.

Mean Opinion Score

Conversely to the PSNR measurement, the *Mean Opinion Score* (MOS) is a **subjective** measurement of quality differences, and is usually obtained by means of a survey and a significantly sized sample of participants. As such, MOS gathering can be time consuming and expensive to perform [25].

In usual cases, participants are asked a number of questions, or provided with some material to watch or listen to. They are required to vote on the question or material in question, typically on a scale of *one* to *five*, signifying different ranges of satisfaction, or agreement, with the question presented. The average score is taken as the MOS.

In Chapter 5, the MOS is used to validate an objectively obtained set of data, based on how frequently the playback of a video pauses, due to buffer a under-run. The reasons why pauses in video playback occur become apparent when TCP is considered, and is discussed in Section 2.3.2.

2.3.2 TCP-based transport

There are two main methods of delivery when TCP is used to transport video. Firstly, a *complete download* requires the entire video to be downloaded to the user's computer before viewing can commence. This can be useful for viewing offline, at a later date, however a long download time may accompany long videos.

The second method is *progressive streaming*, which allows playback to begin after a certain time, usually several seconds. The remainder of the video is downloaded in the background, whilst the video is played back. This is the preferred method, for obvious reasons, but is restricted to compatible video formats such as H.264. In Chapter 5, this kind of video transport is the basis for the research conducted.

Since TCP is a reliable transport protocol, many of the disadvantages discussed for the case of UDP cease to exist. However, with a lossy channel, the issue becomes a restriction in throughput, as a result of the TCP congestion control algorithms. As such, a different kind of quality degradation is noticed.

Quality Degradation

With data recovery an integral part of TCP, any packet loss is recovered by means of a re-retransmission. As a consequence, TCP-based transport experiences none of the issues described in the last section, in terms of quality. Rather, when packet loss occurs, and the TCP window size is reduced, the overall throughput is reduced. As such, the video playback must pause until the data becomes available to recommence playback.

This is an interesting and previously uninvestigated quality issue. In Chapter 5, the issue is explored in depth, and a new quality metric is proposed for objectively measuring this phenomenon.

2.4 Channel Coding

In this section, channel coding will be discussed as a preamble to the research presented in later chapters. *Forward Error Correction* (FEC) is used to protect against data loss during transmission, requiring no further action by the transmitter to recover the data, should loss occur. There are, of course, limitations to how much lost or corrupted data can be detected or corrected, and this will be discussed in detail.
In Chapter 3, FEC is used to protect against packet losses on the Internet, which in turn eases the necessity for TCP to deal with retransmissions. In addition, Chapter 4 uses FEC as a comparative protection technique against the results presented.

2.4.1 Forward Error Correction

In digital communications, in the general case, transmission paths are inherently lossy mediums which require a certain element of protection before error-free communication between two hosts can exist. This is especially true for high-speed or long-distance channels [26]. Different mediums exhibit different loss characteristics, and so require varying levels of protection. Fig. 2.21 shows the *Bit Error Rate* (BER) with and without *Reed Solomon* (RS) codes, and the coding overhead which can be expected with such gains. For example, to achieve a BER of no more than 10^{-6}, a *Signal-to-Noise Ratio* (SNR) of approximately $10.5dB$ is required for uncoded data. However, if RS coding is applied, requiring a 14% overhead, only $6.5dB$ of SNR is required [9]. The Reed Solomon coding scheme will be discussed later.

In addition to the transmission medium, there is also consideration required with respect to the application involved; specifically, how critical the transmitted data is. For example, transmission of data for digital signage should be error free to prevent incorrect characters or information from being displayed. Conversely, however, in some applications, small levels of loss at the receiver can be tolerated, with little or no effect on the final output [27]. This point in particular is one that is drawn on in Chapter 4, where only the most critical elements in a video stream require protection. Moreover, when the unprotected data is subject to packet loss, a high level of playback quality can be maintained.

In simplex communications, that is, communication that occurs in one direction only,

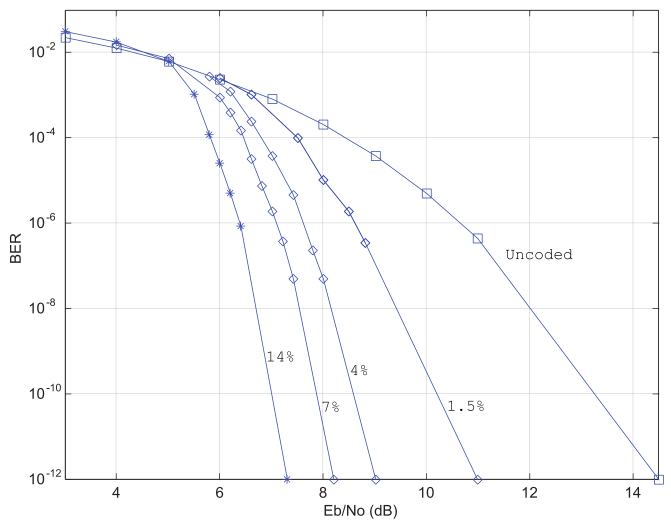


Figure 2.21: Expected BER, measured at the receiver, of a channel with and without Reed Solomon codes. The coding overheads are labelled in each case [9].

FEC is often particularly important because the transmitter has no knowledge of channel conditions. In addition, the receiver cannot provide feedback to the transmitter when data loss has occurred. Retransmission of missing or corrupt data is therefore not possible.
This is especially true for wide-scale broadcast networks such as *Digital Video Broadcast* (DVB) [28], where the channel conditions vary so much between transmitter and geographically different receivers.

For duplex transmission, two-way communication allows the receiving station to signal the transmitter if loss occurs, so that preventative measures can be taken. The most common measures are to reduce the data rate, increase the protection, or increase the transmit power [29]. At higher layers, with respect to the OSI model discussed earlier, there is also the alternative whereby the data is re-transmitted [4]. Re-transmission is a good error- or loss-preventative measure, but is only possible in duplex transmissions or networks. Re-transmission of data is classed as a reactive measure against loss [30], because additional data is only required if loss actually occurs.

FEC itself is a proactive measure against data loss because redundant data is appended to the transmission, which can be used for recovery of information, should part of it become lost or corrupted. In the following subsections, a number of different FEC techniques will be discussed, from simple parity check codes, to more advanced convolutional codes. In particular, Reed Solomon codes are detailed, which are a class of linear block codes used in this research.

In Fig. 2.22, a popular model for digital channels is shown for lossy communications networks. The *Binary Symmetric Channel* (BSC) describes the probability of information reaching the destination on a per-bit basis. Since the bit can only represent two values, zero or one, the probability that it will become corrupt, i.e. change value, is defined as P_e. Conversely, therefore, the probability of error-free transmission is $1 - P_e$.

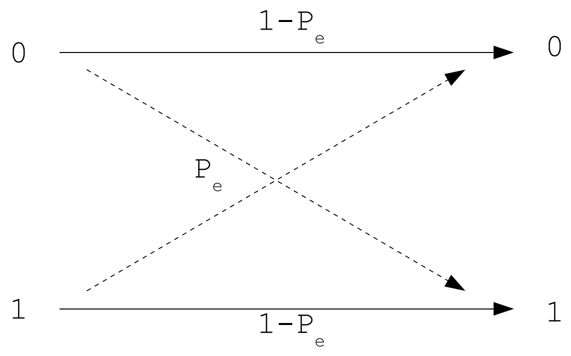


Figure 2.22: Binary Symmetric Channel.

As mentioned in Section 2.1, error detection and frame re-transmission can be performed at the datalink layer of the OSI model, but on Ethernet-based networks, this is limited to detection only [31], since errors are so infrequent [32]. However the specification of other standards such as 802.11 wireless networking [33] makes frame re-transmission a requirement. In either case, the detection of lost or corrupt data exists on all sections of the OSI model, and will be reported to upper layers should recovery be required. If this is the case, it is usual for the transport layer to deal with any losses by re-transmission of the required segment.

The simplest of FEC coding techniques is the Repetition Code. The source coder repeats the transmitted information n times, allowing the receiver to vote on a likely value if one or more repetitions differ. Voting need not occur if all the received values match. As well as being a very inefficient use of bandwidth, the correctional capability is limited. For example, a repetition length of 2 (total length 3) can detect two but correct only one error. The simplicity of decoding, however, is the prime advantage to its use [34].

2.4.2 Parity Check Codes

A parity check code is a codeword which is added to the information being transmitted. It is determined by *Exclusive-OR'ing* (XOR'ing) of all source bits, symbols or packets together (modulo 2 addition). The XOR truth table and demonstration of its operation at bit level is shown in Fig. 2.23.

(a)

A	B	XOR
0	0	0
1	0	1
0	1	1
1	1	0

(b)

```
1001101|0100101|..
10011011|01001010|..
```

(c)

```
1    01001001  (0x49)
2    10011010  (0x9A)
3    10001001  (0x89)
4    00110100  (0x34)
     ------------------
5    01101110  (0x6E)
```

Figure 2.23: XOR truth table *(a)*, and demonstration of operation for: a framed bit stream *(b)*, and a block of bytes *(c)*.

The XOR operator works in the same way as logical OR, with the exception of when both *A* and *B* are 1, the result is 0. This logic provides extremely versatile uses in the coding world [34], not least to produce the primitive parity check code.

The example given in Fig. 2.23 provides two demonstrations of parity check operation. In *(b)*, a framed k-bit sequence is XOR'ed, in accordance to truth table (a), to give an additional parity bit and a code length of n. The frame boundaries of the codeword are shown. This allows detection of transmission error should an odd number of bits become corrupt (change state). Example *(c)* shows a block of 4 bytes, represented in binary and hexadecimal form. The bits from each column are XOR'ed to produce parity as a fifth byte. If an odd number of bits are changed during transmission, the error can be detected but not corrected, since the location of the error is not known. Should one of the bytes be lost during transmission, it can be reconstructed using the three existing information bytes XOR'ed with the parity byte.

The *distance*, d, between two codewords provides a measure of error correction capability. Expressed in this way, a coding scheme can detect $d-1$ errors and correct $(d-1)/2$. The distance between two codewords can be determined by examining the difference between each bit, or mathematically by the sum of all adjacently XOR'ed bits. For example, to find the distance of codewords 101_2 and 110_2, $d(101,110)$:

$$d = (1 \oplus 1) + (0 \oplus 1) + (1 \oplus 0) = 2 \tag{2.1}$$

2.4.3 Block Codes

Parity check codes are a special case of block codes. With linear block codes, k information bits are taken, placed into a message block, m, and then mapped to a codeword, c, of length n. The additional $n-k$ bits provide redundancy for error detection or correction. The messages block, combined with parity data, is called an (n,k) code. Using this scheme there are a possible 2^n sequences which can be produced. From the entire 2^n possible sequences, 2^k codewords are generated for a complete mapping of k-length blocks. The generation is not arbitrary, however. The codes must be suitably *distanced* from each other to maximise the capability of the code.

If a $(7,4)$ linear block code is taken, typically referred to as a Hamming code, the number of n-length codewords which need producing to represent all k-bit combinations is $2^k = 16$. Since the length of the codeword is n, there are $2^n = 128$ sequences to choose from. To find a set of linearly independent codewords, the easiest method is to select sequences which have only a single 1 in them for the first k positions and then, for the remaining $n-k$ positions, select codewords which have a single 0 [35].

A *generator matrix* is used to produce codewords for all possible sequence combinations of message m, which in this case is 16.

$$\mathbf{G} := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{2.2}$$

Given the generator matrix in Eq. 2.2, for a $(7,4)$ Hamming code, codewords can be produced by multiplying with the original message block, m, such that codeword $c_i = m_i\mathbf{G}$.

Alternatively, considering a message block $m_{13} = 1101_2$, it is seen that bit positions 1, 2 and 4 are set. The codeword c_{13} can therefore be produced by adding the first, second and forth rows in matrix $\mathbf{G}$ using modulo 2,

$$\begin{aligned} c_{13} &= \mathbf{g_1} \oplus \mathbf{g_2} \oplus \mathbf{g_4} \\ &= 1000110_2 \oplus 0100101_2 \oplus 0001111_2 \\ &= 1101100_2 \end{aligned}$$

We see the first part of the matrix in Eq. 2.2 is an identity, which means that any message which is encoded with this generator will result in a codeword which begins with the original message, followed by 3-bits of parity. The code rate, r, of this scheme is given by $k/n = 4/7$.

The example given is a case of systematic coding, since the input message is part of the coded output. This allows a more efficient decode if there are no errors detected. Moreover, if too much data is corrupted in order to correct the block, remaining parts of the message are are still available for salvage [36].

The generator matrix can be modified in order to produce non-systematic codes with the same distance, and hence detection/correctional capability [37] [38] using column permutations and the addition of elementary rows [34]. Non-systematic codes hold no resemblance to the original message, m, and therefore parts of the message cannot be salvaged, as for the case of systematic codewords.

A subset of linear block codes are referred to as cyclic codes, due to their simplistic algebraic structures. Despite this, cyclic codes maintain a strong error correction capability [36].

BCH codes are a further subset of multilevel cyclic codes, which allow for efficient encoder and decoder design [35]. In a later section, Reed Solomon codes are discussed, which are a special case of BCH codes based on cyclic linear block codes.

2.4.4 Convolutional Codes

A convolutional code is such that the encoder remembers a number of previous bits or frames for use in the encoded output [35]. A bit-level convolutional encoder can be easily represented in hardware using one or more shift registers. The input to each register can be either the information bits directly (in which case the encoder produces non-recursive codes), or there can be an element of feedback from other registers (producing recursive codes). An example of a convolutional encoder is shown in Fig. 2.24.

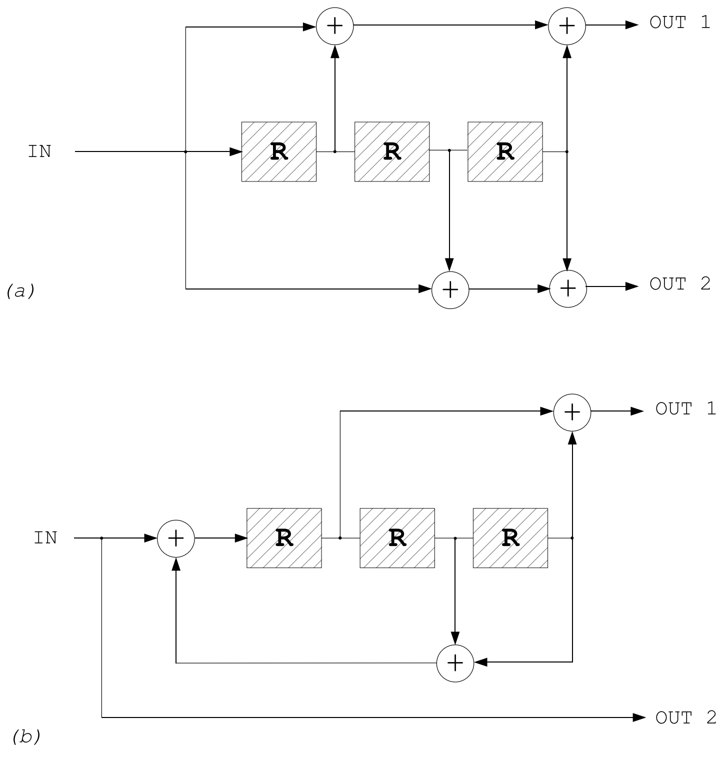


Figure 2.24: A arbitrary convolutional encoder using shift registers for non-recursive codes *(a)*, and recursive codes *(b)*.

The inputs of both encoders in Fig. 2.24 are shifted into two simultaneous outputs, giving a code rate, $r = 1/2$. Each register will shift data once per clock cycle, with the *memory size*, m, of the system defined as the maximum number of cycles between input and output, in this case 3. The constraint length of a convolutional code, K, and is given by $K = m + 1 = 4$. More complex encoders can be produced using more inputs and outputs. There are many high-performing convolution codes available, found in relevant literature [35].

A Viterbi decoder provides means to achieve *Maximum Likelihood* decoding of the received codeword [39]. The complexity of the decoder becomes more complex in this case and, as such, an increase in logic is required.

The decoding process initially involves determining the distances between the received codeword and every other codeword used in the coding scheme. There is then a *hard* or *soft* decision made as to the likelihood of bit corruption. In the case of a hard-decision, the Hamming distance is used as a metric, as discussed earlier. For soft-decision, the decoder

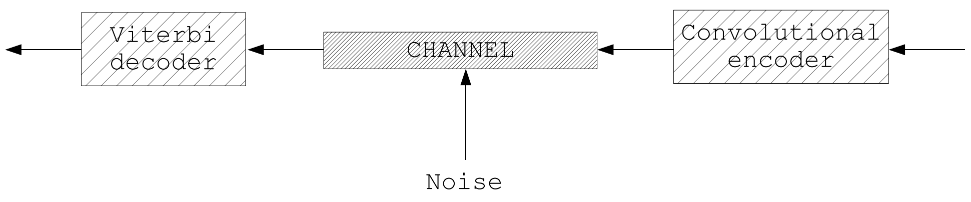


Figure 2.25: A Viterbi decoder within a convolutional coding system.

uses information in the bit stream to determine the reliability of the data. In both cases this is followed by a path metric selection for all 2^k paths of the codeword, that is, all possible path permutations of corruption that the codeword may have taken. From this an optimal estimation is made [34].

2.4.5 Reed Solomon and Erasure codes

Up to now, transmitted data has only been considered as a series of framed bits. However, as discussed in Section 2.1, when dealing with data on a packet-switched network, packets which have become corrupt are not passed up the OSI stack. Rather, they are discarded at the lower layers and, in the case of Ethernet, left for higher layers such as the transport layer to deal with. The transport layer sees these errors only as entire missing packets, which are termed Erasures. As such, one must consider special coding techniques, such as *Reed Solomon* (RS) codes, which are a special non-binary case of BCH codes, and based on cyclic linear block codes. RS codes have been chosen because of the wide range of encode/decode designs available, which is important when considering the hardware-based implementation discussed in Section 3.6.

Erasure codes are a form of FEC used for communication though a lossy medium. When decoding the encoded data using Erasure codes, the receiver is assumed to know the exact location of the lost packets. This information is not needed in a general FEC technique, since correction is done on blocks of bits, as discussed previously. Erasure codes are typically used for sending packets through the Internet, since the receiver can detect the location of the lost packets by noting the missing packet sequence number. Knowing the location of an error or Erasure is important knowledge for successful recovery [35].

The operation holds a similar methodology to that of the binary case. In a typical Erasure

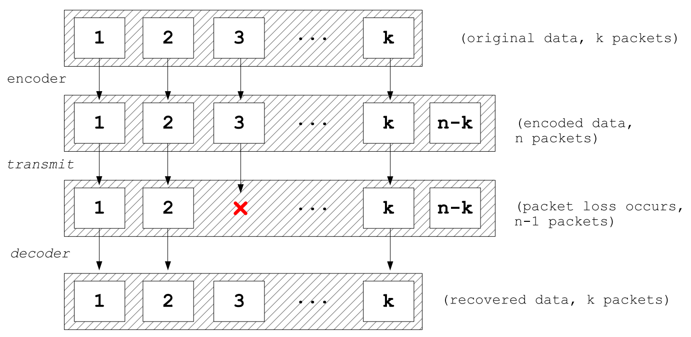

1
2
3
...
k
(original data, k packets)
encoder
n-k
(encoded data, n packets)
transmit
(packet loss occurs, n-1 packets)
decoder
(recovered data, k packets)

Figure 2.26: Reed Solomon codes for packet-based coding.

code, the sender encodes redundant packets before sending both the original and redundant packets to the receiver. The receiver can reconstruct the data upon receiving a fraction of the originals packets. Standard Erasure codes, such as the RS (n,k) Erasure codes, takes k original packets and produce $n-k$ redundant packets, resulting in a total of n packets. If k or more packets are received, then all original packets can be completely reconstructed, as shown in Fig. 2.26. Hence, a larger n/k ratio leads to a higher level of protection for data [40].

The Reed Solomon code is optimal in the sense that the minimum distance has the maximum value possible for a linear code of size (n,k); this is known as the Singleton bound. Such a code is also called a *Maximum Distance Separable* (MDS) code. RS codes as MDS codes have been suggested for application of packet loss protection in notable research [41], including for specific cases of transporting video [42] [43].

Galois field (or finite field) arithmetic is fundamental to RS encoding and decoding. Each field has a finite number of elements (often expressed as polynomials) which, when operations are performed upon, take on another element in that field. There are explanations and several approaches to performing Galois field multiplication in software here [44] [45].

Polynomial representation of codewords can be used by expressing each binary value as increasing powers of α, for example $000 = 0$, $001 = \alpha^0$, $010 = \alpha^1$, $100 = \alpha^2$, $101 = \alpha^3$ etc. This allows easy addition of finite field elements, since polynomial addition is trivial.

Given a data polynomial $a(x)$ of degree p, where $p <= n-k$ and

$$g(x) = \prod_{i=0}^{p-1}(x+\alpha^i), \alpha(x) = \sum_{i=0}^{k}\alpha_i x^i \tag{2.3}$$

with α^i successive unity roots in $GF(2^m)$ and α_i elements of the same field, the systematic encoding of $\alpha(x)$ is given by

$$C(x) = \alpha(x)x^{n-k} - R(x) \tag{2.4}$$

where $R(x)$ is the remainder of the division $\alpha(x)x^{n-k}$ by $g(x)$.

RS coding offers optimal efficiency such that any available parity element can be substituted for any erased data element in the block. The computation costs of this process is related to the size of the field, where typical RS code implementations operate in a field of size 2^8, or one with 255 elements.

2.5 Conclusion

In this chapter, several key background topics have been discussed, which are relevant to the research conducted throughout this Thesis.

Firstly, a detailed overview of the IP network infrastructure was presented, discussing the most relevant transport protocols and the problems that are associated with each. For the case of UDP, losses cannot be controlled. For TCP, integrated flow control ensures packets are delivered, even if loss occurs. This, however, causes the throughput to drop significantly as the transmitter assumes a congested network, which triggers a reduction in the window size. The layers of the OSI model were discussed in detail, including the transport protocol, for which TCP and UDP are based.

The following section provided useful discussion on the requirement for video compression, outlining the history of generalised compression techniques such as chrominance subsampling, intra- and inter-frame coding, run length encoding and Huffman coding. The concept of a GOP was introduced, which contains spatially compressed frames (I-frames) or temporally dependant frames (P- and B-frames). In addition, the enhancements provided by the recent H.264 standard are discussed, which include a NAL and the ability to split the video stream in the partitions of more or less importance. This is a key point that is used in the research in Chapter 5. The compression efficiency of H.264 was also demonstrated by comparison to the popular MPEG2 coding standard, and found to be significantly better.

The flow control features of TCP, and lack thereof for UDP, are combined using the background discussed in the first two sections to present background on video streaming itself. Having determined that packet losses cannot be controlled when using UDP, several key points are made with respect to quality degradation, and objective methods of measuring this loss of picture information, namely using PSNR.

Since TCP is able to control losses, the quality degradation manifests itself as a different property - breaks in the continuity of playback. The measure of PSNR, therefore, does not apply. The objective measurement of pause intensity is introduced as an alternative, which is the basis for the research conducted in Chapter 5.

In addition to objective measures of quality, the MOS is outlined, since it is used later to validate the effectiveness of the newly introduced quality assessment technique, pause intensity.

The last section discusses background on channel coding, in particular for the RS family of codes which can be successfully used in packet-based, best-effort networks to protect against loss. Chapters 3 and 4 draw heavily upon this background.

Chapter 3

Combining TCP with FEC

This chapter will look at the benefits and limitations of content distribution using the *Transmission Control Protocol* (TCP) with *Forward Error Correction* (FEC). A practical approach is presented for combining TCP with FEC as a loss recovery procedure, taking into account resulting overheads and throughputs at different rates of packet loss. The purpose of this research is to emphasise the importance of balancing the two technologies when used in a practical situation. To ensure the efficient combination of both adding redundancy and requesting retransmissions, adaptive coding will be the primary consideration.

3.1 Introduction

FEC can be used to reduce the number of retransmissions which would usually result from a lost packet. The requirement for TCP to deal with any losses is then greatly reduced, since data is recovered, negating the requirement for retransmission. There are, however, side-effects to using FEC as a countermeasure to packet loss: an additional requirement for bandwidth. When applications such as realtime video conferencing are needed, delay must be kept to a minimum, and retransmissions are certainly not desirable. A balance, therefore, between additional bandwidth and delay due to retransmissions must be struck.

FEC is most notably used for Digital Video Broadcasting (DVB) [28] [46] using satellite or terrestrial antennas [47] and digital music storage on Compact Discs [48], both coded at bit-level, where buffering is minimal and the flow of data is one-way (simplex). High levels of FEC are required due to the lossy characteristics of free-space transmission and storage mediums prone to damage. These existing coding schemes have been reused for other technologies such as *Asymmetric Digital Subscriber Line* (ADSL) [49] with great success.

Recently researchers have looked into combining this with packet based networks such as the Internet, in particular when the same kind of delay constraints as video broadcasting are imposed. End-user applications such as video conferencing and telephony can often benefit from FEC, and with bandwidth limitations relaxing, more powerful FEC can be applied to ensure that losses in transmission can be minimised.

The performance of TCP has been a major area of research over the past decade with some very accurate models produced which analyse the throughput variations when packet loss occurs [50]. They have found that session timeouts are much more common than anticipated with long-lived TCP connections, and can have great effect on the average throughput of a channel. It is also seen how the performance of TCP is greatly reduced when the last hop of a connection is wireless. As [29] has shown, one must increase either the transmit power of the wireless physical layer, the level of FEC protection, or the number of retransmitted packets, as a consequence. Since many home users now make use of a wireless last-hop, the original specification of TCP may need to deal with a new pattern of packet loss and delay statistics, despite the use of 802.11 frame retransmissions at the data link layer. Existing research has also been done to investigate the positive uses of FEC within the Internet for improving performance, in different applications ranging from file sharing [51] to video streaming [52].

However, the draw backs to using such techniques are often left under-detailed, or simulations are done which unfaithfully show benefits without side-effect; an issue which this research addresses with strictly conducted experiments. This work is centred around a real world application looking at both the performance benefits and drawbacks to using FEC on the Internet, since much of the time the TCP window size is not at its maximum when packet loss occurs [53]. Similar research has shown how FEC can protect TCP [54], and concluded that this comes at the expense of requiring additional bandwidth, a point akin to this research criteria. This research builds upon the work done by others by running real network tests and applying the results to show improvements to video transmission performance.

Section 3.2 will outline some preliminary background research which has been considered during this work, covering the basis of *Reed Solomon* (RS) coding [55] and how this can be combined with TCP. It also sets the metric for why this research has been carried out. Section 3.3 explains how the results were collected, and in Section 3.4 the collected data is analysed, making clear the significant results. Section 3.5 is a case study to demonstrate how the research can be applied practically, and illustrates the improvement in performance. Section 3.6 investigates a network solution for decoding FEC-protected packets using a hardware-based FPGA design, and discusses the potential this yields for future work. Finally, conclusions are drawn in Section 3.7.

3.2 Preliminaries

To model random packet losses, the Bernoulli model is considered. The details of this, along with a definition for *residual loss rate* is presented in Section 3.2.1. Section 3.2.2 details on the background to applying FEC in IP-based networks, and Section 3.2.3 explains how this is combined specifically with TCP.

As a preamble to the experimental procedure and results, Section 3.2.4 reaffirms the research methodology, aims and objectives.

3.2.1 Packet loss model

For these experiments, randomly distributed packet loss is used, which is based on the Bernoulli model. The Bernoulli model is used to model all possible outcomes, given a set of expected values - in this case, the *success* or *loss* of a packet transmission. The model has a sample space, Ω, containing two possible outcomes: s, for when the packets are received *successfully*, and l, for when packets are *lost*. The sample space can therefore be described as $\Omega = \{s, l\}$. If a random variable, X, is defined on to this sample space such that $X : \Omega \mapsto \mathbf{R}$, then X is given by

$$X(\omega) = \begin{cases} 1 & \text{if } \omega = s \\ 0 & \text{if } \omega = l. \end{cases} \tag{3.1}$$

If P is the probability packet loss and $1 - P$ the probability of packet success, or when defined using the model, 1 and 0, and these values exist in the probability distribution of X, then X has a Bernoulli distribution. The mean (expected value) and the variance of X are given by

$$E[X] = 1 \cdot p + 0 \cdot (1-p) \ = p \tag{3.2}$$

$$V[X] = E[X^2] - E^2[X] \ = p - p^2 \ = p(1-p) \tag{3.3}$$

Residual Packet Loss, P', is defined as the packet loss rate after a coding scheme has been applied. For example, if the packet loss probability, P, is 0.05 then on average 5% of all transmitted packets will be lost in transit, but if 20% of those packets are corrected due to a coding scheme, the amount of data lost will decrease to 4%, or $P' = 0.04$.

With a $(7,5)$ RS code, it is required that at least 5 out of the 7 transmitted packets are received to maintain a 0% residual loss rate. To determine the probability that an entire block of 7 packets will suffer up to two losses, the probability of loss for each packet must be considered. An illustration of this is shown in the tree diagram of Fig. 3.1, where the probability of successful delivery is $1-P$.

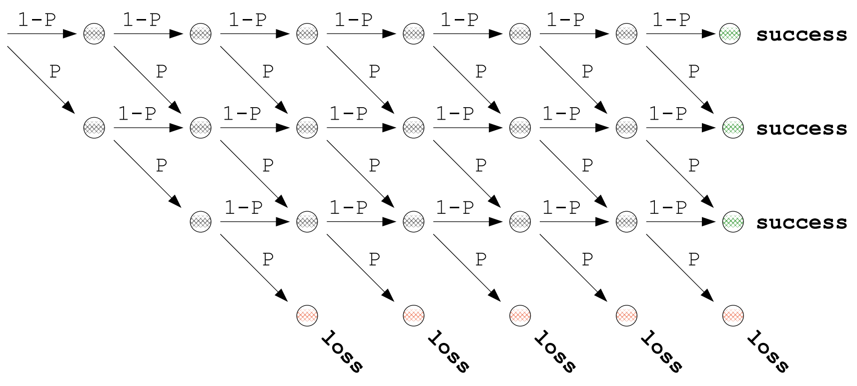


Figure 3.1: Block loss probability tree diagram for a $(7,5)$ RS coding scheme.

For the complete derivation of this model, the probability of error free transmission for a single packet must first be considered, and then extended to a block of n packets. If P is the probability of a single packet loss, then the probability of error-free transmission is $1-P$ per packet. It follows, then, that the probability of error free transmission, P_{free}, for a block of n packets is:

$$P_{free} = (1-P)^n \tag{3.4}$$

Now consider the probability that only *one* packet in a specific position within the block, n, is transmitted error free:

$$P_{one} = P(1-P)^{n-1} \tag{3.5}$$

To consider every permutation of this model, positions $1..n$, the probability of failure, due to the number of losses between $(n-k+1)$ and n, must be summed. In each case every possible path should be considered, as illustrated in Fig. 3.1; or in other words, to consider a single probable loss at every packet position within the block:

$$P_1 = \binom{n}{1} P(1-P)^{n-1} \tag{3.6}$$

It follows then that for all losses within the block n,

$$P' = 1 - \sum_{i=0}^{n-k} \binom{n}{i} P^i (1-P)^{n-i} \tag{3.7}$$

which is equivalent to:

$$P' = \sum_{i=n-k+1}^{n} \binom{n}{i} P^i (1-P)^{n-i} \tag{3.8}$$

Using this, the residual loss rate, P', after recovery by an RS coding scheme, can be modelled. The parameters required are n, k, and a given packet loss rate, P.

3.2.2 Erasure coding

When applying FEC to Internet traffic, the IP stack will only present entire packets upwards to the network layer, and if corruption occurs at a lower layer the packet will be dropped [5]. These losses can be considered as Erasures to which FEC can be applied [56]. In order to apply FEC to transmitted data, packets must first be grouped into blocks, some of which will be the generated redundancy. By adapting the coding algorithm to deal with data in a per-packet manner, this capability can be applied to networks modelled on the Open Systems Interconnect (OSI) [5]. Erasures can then be thought of as the corruption of a group of packets, or a *block* herein.

Reed Solomon (RS) codes [55] are a group of *Maximum Distance Separable* (MDS) codes, which means that they meet the Singleton bound, $d = n - k + 1$, and are able to correct $n - k$ losses in a block of n symbols, where k is the number of data symbols and d is the minimum distance of an RS code. The units of data can be symbols, bytes, or in this case, whole packets.

In lossy networks, adaptive FEC can be used to maximise the efficiency of the coding scheme and the throughput. When low levels of packet loss are experienced, minimal redundancy is required to recover data loss. Conversely, at high rates, a much greater level of redundancy can be appended.

As discussed in Section 2.4, there are two coding options - systematic and non-systematic - which allow for the original packets, $p_1, p_2 ... p_k$, to be encoded into a blocks $b_1, b_2 ... b_n$. In the non-systematic coding case an entire set of packets, $p_1, p_2 ... p_k$, is encoded in such a way that the resulting block, b_E, has no similarity to the original packets. Only after successful decoding can the original packets be restored. As was shown in Fig. 3.1, if more than two encoded units within b_E are missing, the entire block can be considered lost. In systematic coding, $b_1 = p_1 ... b_k = p_k$, meaning the original packets remain unaltered when coded into block units, and the redundancy is appended to the existing data. If more than two packets are lost using this method of coding, it may be possible to recover fractional components of the encoded block [56]. It can therefore be stated that when using a non-systematic coding scheme, the probability of block loss is equal to P', the residual packet loss defined in 3.2.1, because either all or none of the transmitted data can be recovered when using this method of coding, as discussed in Subsection 2.4.3

3.2.3 Combining FEC with TCP

Having outlined the method to correct blocks where $n - k$ or fewer packets are lost, the cases where too many packets are lost must also be considered. This is where the *Automatic Repeat Request* (ARQ) functionality of TCP comes into use, and retransmissions become a necessity to ensure reliable delivery. Fig. 3.2 shows the five primary layers of the TCP/IP stack, which are based on the OSI model [14], and discussed in Section 2.1.

The problem behind TCP for many applications is the sudden decrease in the window size when congestion is detected. Despite there being very valid reasons for this amount of rate adaptation [57], there are many alternative versions of TCP which look to modify this behaviour for reasons of improving performance. This makes time-critical applications such as video streaming more reliable. This research aims to take this a step further by introducing FEC into the process of requesting a retransmission, and adapting the level of protection depending on the level of packet loss. When dealing with video transport, it makes little sense to blindly halve the data rate of a channel, for example, if the data rate of the video remains constant. This will significantly affect the quality of playback perceived by the viewer. Instead, one can reduce the data rate of the video, if necessary, and allow further protection of the stream using stronger FEC, which will better balance the throughput and success of block correction. The data rate will still be reduced, but not so drastically as to cripple performance. This allows applications such as video transmission to be better protected against breaks in the fluidity of transmission.

The example given in Fig. 3.2 shows three blocks of packets, (a), (b) and (c). Each have been encoded using a $(7,5)$ coding scheme. In case (a), none of the seven packets have been lost and correction is not required. TCP passes the packets directly to the application layer, specifically the running *iperf* program, where the data is accepted.

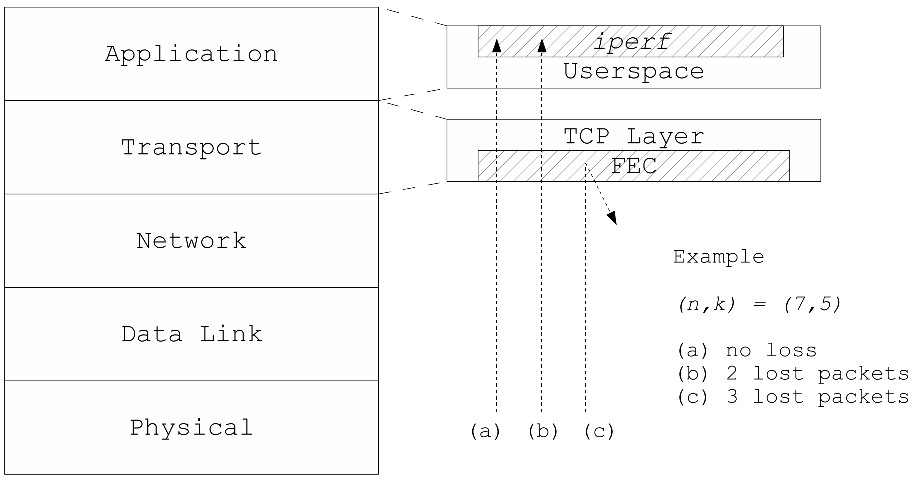


Figure 3.2: The TCP/IP stack with FEC addition, and a theory of operation example.

In case (b), two packets have been lost within the block during transmission over the network. The FEC layer is able to recover these losses using data from the other packets received. The data is then passed to the TCP layer where it appears no loss has occurred, which in turn passes the data to *iperf* as in the case of (a). In case (c), the loss of three packets means FEC is unable to recover the lost packets and must rely on TCP to trigger a retransmission, until *k* packets have been received, at which point the correction can occur.

3.2.4 Research Methodology

Before the experimental details and results are presented, the research methodology is made clear for this work. The aims and objectives are outlined below.

Aims

Rapid reduction in throughput, as shown in Fig. 3.3, is the primary concern in this research. This occurs when packet loss is experienced on the channel, and is a result of the TCP congestion algorithm. Under normal circumstances, this rapid decrease in throughput is not noticed and, at most, may cause a webpage to take a few seconds longer to load.

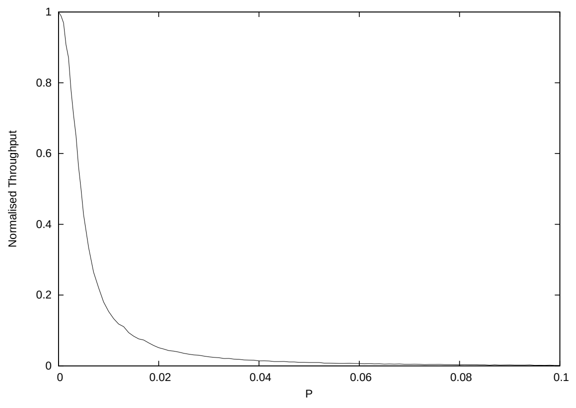


Figure 3.3: Normalised TCP throughput experienced when the channel is subject to packet loss [10]

When video transport is involved, however, the reduction in throughput has a greater effect due to a high dependence on available bandwidth. It is aimed to address the severity of the problem by introducing adaptive-rate FEC, before TCP, to lessen the reliance on retransmissions, guarantee delivery, and therefore maximise the available window size of TCP. Through a case study applying specifically to video transmission, the benefits are made clear.

Objectives

- Conduct a strictly controlled set of tests to investigate the problem of rapid TCP throughput reduction.
- Demonstrate, through experimental result, the benefits to using such a system.
- Clearly highlight the limitations to using the proposed system.
- Apply the results to a case study to demonstrate the notable advantage when used with video streaming.

3.3 Experiment Setup

A TCP stream was established using Linux utility *iperf*, and throughput measurements taken at varying packet loss rates. Loss rates were controlled by a second networking tool, *tc*, within the range 0%-10% and with a 0.1% step size. The data was transmitted on the loopback interface to ensure that all losses were controlled. For $0 \leq P \leq 0.005$ the step size was reduced to 0.05% to improve accuracy at the early stages of TCP congestion control. The structure of the test bed is shown in Fig. 3.4 as two constituent parts. One to collect the throughput statistics, and a second to monitor the bandwidth overhead due to re-transmissions. Both parts run simultaneously.

To see how adding FEC affects throughput, Eq. 3.8 is used to calculate a residual packet loss rate for each value of P. The experiments are repeated using the P' values (loss experienced after packet recovery), but plotted against P (actual loss on the channel). This gives a series of comparable throughput curves for $0 \leq P \leq 0.1$ using different levels of redundancy.

The methodology of the data recovery process is thus: when packets within a block are lost, the decoding process will be able to recover the lost packets if the number of losses is no more than $n-k$, otherwise the lost packets will have to be recovered through the TCP retransmission mechanism. This is because all the MDS codes can correct up to $(d-1)$ or $(n-k)$ Erasures, which is guaranteed by the Singleton bound.

Recording the bandwidth overhead was achieved using the Linux module *tcp_probe* which provides a *hook* into the lower layers of the IP stack. This is particularly useful

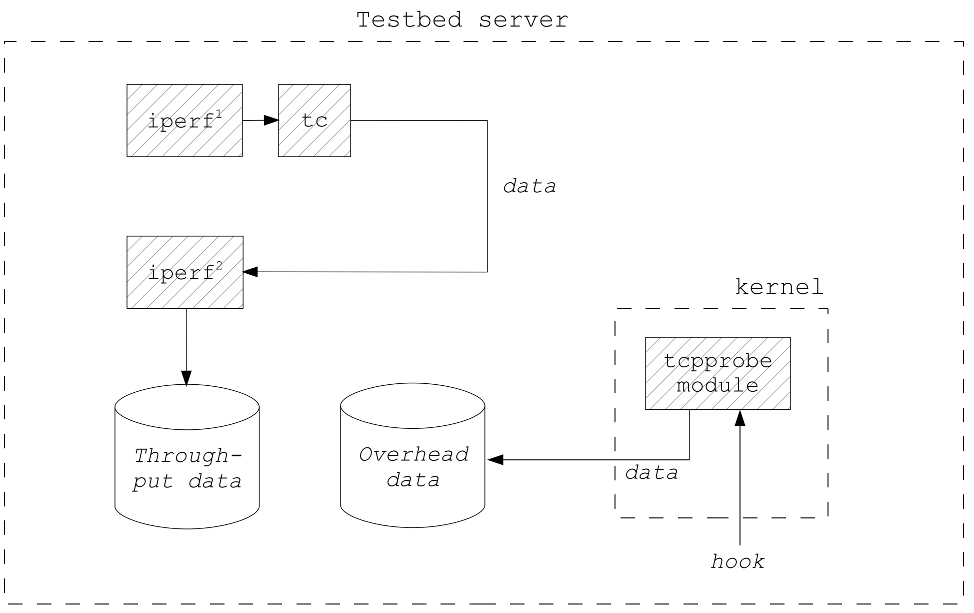


Figure 3.4: System block diagram of the test bed setup. ^{1}Server and ^{2}Client

for monitoring the state of TCP endpoints and can be accessed using the */proc/net/tcpprobe* pseudo filesystem. An output example of the *tcp_probe* module is shown in Listing 3.1.

Listing 3.1: tcp_trace output format

```
5.617786188  192.168.2.4:80  192.168.2.191:1877
450  0xf2880cf8  0xf2880bed
3  2147483647  17640  0
```

The output prints one line per packet. In this example, the data has been split into three lines for clarity. The first line begins with a *timestamp*, in seconds, from when the probe began. The next two fields are the *source address* and *destination address*, along with their associated port numbers, after the colon *(:)*. The second line begins with the *number of bytes* within the packet, followed by the *next sequence number* and finally the *unacknowledged sequence number*. The third line provides the *congestion window*, *slow start threshold* and the *send window*. The *number of bytes* field is used to calculate the total bytes sent, including any retransmissions - something which the *iperf* does not provide. This is the primary benefit to using *tcp_probe* in this research.

Linux kernel 2.6.23 was used, which provides TCP Cubic [58] by default. *iperf* v.2.0.4 was used which allows for a *Comma Separated Values* (CSV) output. The machine used had two dual-core Xeon processors running at 3.46GHz and 1GB of RAM. Swap space was not utilised.

3.4 Results and Analysis

Results were initially taken without any FEC protection, which are shown in Fig. 3.5. As expected, with the congestion control algorithm of TCP Cubic, as packet loss increased more retransmissions were required.

Unsurprisingly, it is noticed that when $P = 0$ there is no degradation in throughput, and the window size remains at its maximum. However, when losses are introduced, the window size dramatically decreases and a very noticeable reduction in throughput is experienced. For instance, a packet loss rate of only 0.5% $(P = 0.005)$ results in a reduction of around one half the throughput when compared to the loss-free case.

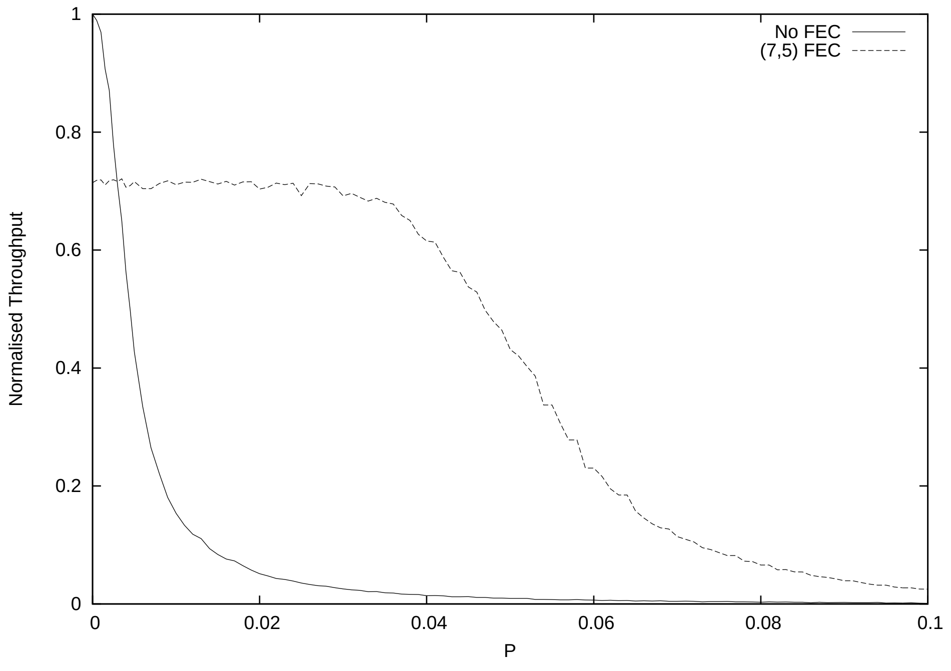


Figure 3.5: Throughput of a standard TCP Cubic stream, and that of an RS-coded stream.

This over-zealous behaviour is designed to prevent congestion problems by reacting quickly to any requests for packet retransmission. However, in the case of video transmission this can have crippling performance consequences if a constant rate of data is required, or if the video data rate is close to that of the user's available bandwidth. This behaviour and sudden throughput reduction was the motivation behind working toward a solution which was better suited to video transport, whilst maintaining the congestion control algorithms inherent with TCP.

The results were taken once more, but this time with some additional redundancy data included. Values for n and k were chosen as $n = 7$ and $k = 5$ which amounted to an increase in bandwidth of nearly 29%. With the coding implemented, an instant improvement was noticed. The residual packet loss, P', remained close to zero until $P = 0.04$, giving an instant improvement in the overall throughput for packet loss rates less than 4%. As a consequence, throughput remained approximately constant up to this level of loss. To account for the additional bandwidth required by the redundancy, the measured throughput was reduced by $2/7$ so as to be comparable with the TCP-only tests, and this is reflected in the $(7,5)$ curve of Fig. 3.5.

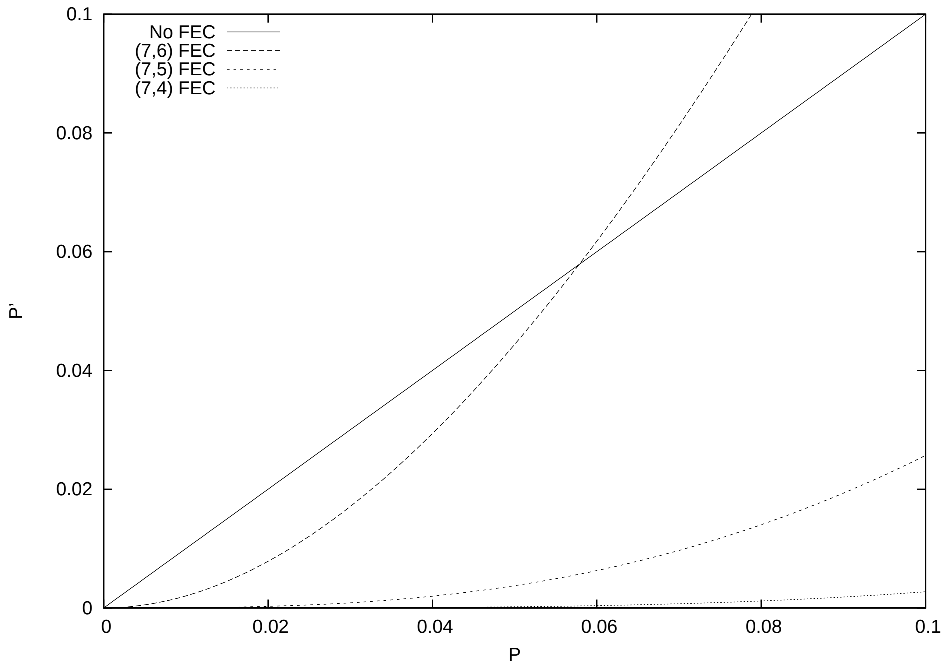


Figure 3.6: Residual packet loss (post correction).

Fig. 3.6 shows the values of P against P' for four FEC scenarios, calculated using the analytical derivation at Eq. 3.8. Due to the non-systematic nature of the coding scheme, it can be seen how, for the case of No FEC *(TCP only)*, $P = P'$ (see Section 3.2). The point at which the RS coded results intersect the *TCP only* curve is where the addition of redundancy causes $P' > P$, signifying an increase to the residual loss rate compared to the original loss. This occurs because the loss of too many packets is causing a decode failure for the entire block, and all remaining packets are discarded as per the non-systematic case. This is obviously an undesirable situation, since the rate of loss, P, is actually lower. If the case is considered where $P = 0.04$ (4% loss), the residual loss rate P', that is, after correction, can be reduced to approximately 0.03 (3% loss) when using a $(7,6)$ RS code. It is also noticed that when the value of k is changed, a dramatic effect is had on the capability of the coding scheme, as shown in Fig. 3.7 where three coding schemes are tested. Adding a further packet of redundancy per n packets, using the $(7,4)$ code, shows an even greater improvement over the case of the $(7,5)$ code in terms of throughput sustainability against a wider range of loss. At this level residual losses remain close to zero even at loss rates around 10% $(P = 0.1)$, in exchange for a reduction in throughput due to additional bandwidth requirements. Even for a $(7,6)$ code, which is a simple parity check code, improvement is noticed for values of P below 0.045, although loss is still experienced. This demonstrates how powerful these coding schemes can be and illustrates the potential need for coding in video streams.

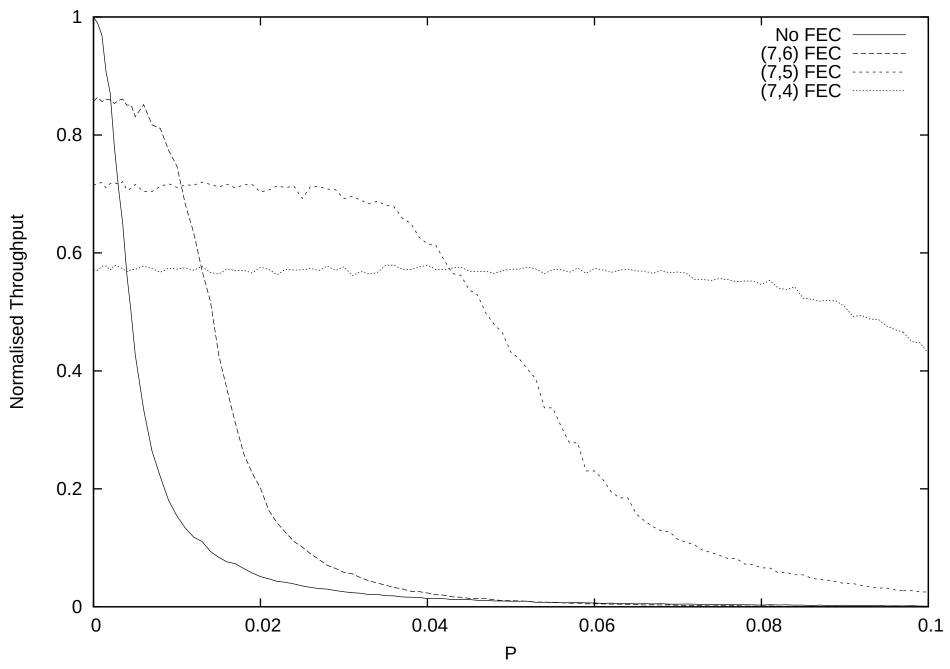


Figure 3.7: Performance of other coding levels in terms of throughput.

These tiered levels of protection using different coding schemes, depicted by the combined curve envelope in Fig. 3.7, are an important result when it comes to ensuring video is transmitted in an efficient manner. For any given loss rate, P, the throughput can be maximised by using the coding scheme which forms part of this envelope. For example a loss rate of 2% (P=0.02) is shown to have a maximised throughout when using a $(7,5)$ coding scheme; 6% (P=0.06) should use $(7,4)$. The crucial requirement when transmitting video is that the rate at which data arrives remains constant, so that the bit rate and resolution of the video can then be modified to suit, and adapted if necessary. The data from this result is used for the case study in Section 3.5.

When packet loss does not occur, there is simply no need to include redundancy within the block of data transmitted, and so the throughput can be maximised.

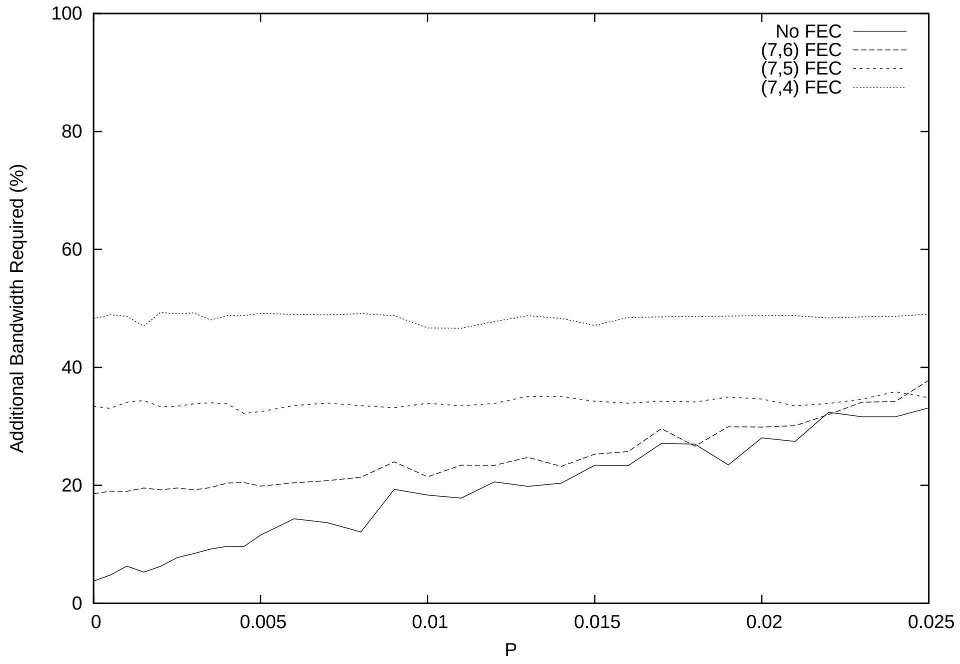


Figure 3.8: The bandwidth overhead required to ensure successful delivery over a range of P for FEC and *TCP only* streams.

Fig. 3.8 illustrates the initial overhead required for each case of coding, and for *TCP only*. For the *TCP only* case, the overheads are initially low (just TCP/IP headers) and a uniform increase is noticed as more retransmissions are requested. When redundancy is added, a larger level of overhead is required even for when $P = 0.0$; over 30% for the case of the $(7,5)$ code. This includes both added redundancy and the protocol overheads. It is also noted that for each case that includes redundancy, there is an initial period whereby the bandwidth remains relativity constant for low values of P. This signifies the successful correction of incoming blocks, and only when these recoveries begin to fail does the bandwidth increase once more due to retransmissions. As well as ensuring that the throughput remains high, one must also consider the consequences of adding redundancy, especially if the network is already highly congested; a sudden jump in bandwidth requirements could cause more harm than good. To counter this, dynamically reducing the data rate of the video should be considered.

3.5 Case Study

As set out in the aims and objective to this research, a particular example is considered in this section to demonstrate how the results can be applied to a practical application. Since the system uses TCP as the transport protocol, it is known that the video data is guaranteed to arrive at the client, unless the connection is closed. There would be no degradation in video quality since any unrecoverable losses within the data will be retransmitted. However, the process of retransmission adds significant delay to the arrival time of the data and may cause the entire picture to pause if the local playback buffer is exhausted. This is certainly undesirable from the viewer's perspective, since the video will become unwatchable if the buffer continuously empties or even ceases to recover. This buffer behaviour is further discussed in Chapter 5.

Loss Rate (%)	FEC scheme (n,k)
4.3 – 10.0	(7,4)
1.1 – 4.3	(7,5)
0.25 – 1.1	(7,6)
0.0 – 0.25	*none*

Table 3.1: Table of FEC values showing the best coding scheme for a given loss rate, taken from the maximum-throughput envelope of Fig. 3.7

If a realistic video data rate of $500kbit/s$ is taken, the channel bandwidth set to $1Mbit/s$ and the case where there is no FEC is considered, every lost packet will have to be retransmitted and the throughput reduces rapidly in accordance to Fig. 3.7. From Fig. 3.9 and Table

3.1, it is seen that when the loss rate exceeds 0.25% ($P = 0.0025$), the throughput is below the rate required to sustain the video data rate and the playback buffer will start to reduce. If this situation is sustained, the buffer will empty and cause the playback to pause.

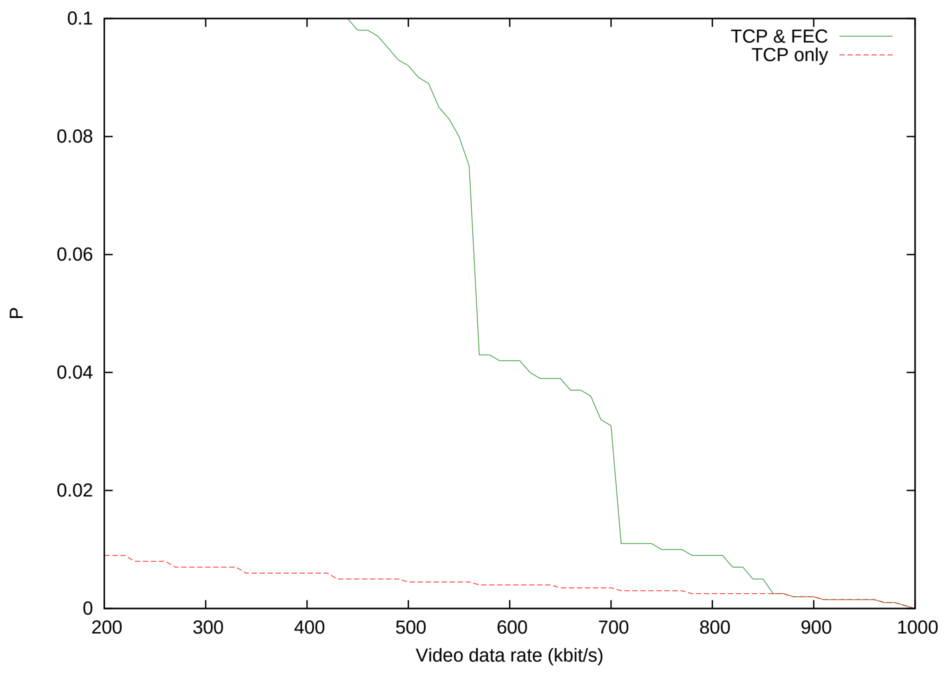


Figure 3.9: The maximum sustained loss rate allowable for a range of video data rates.

Next, adaptive FEC coding using the $(7,4)$, $(7,5)$ and $(7,6)$ codes, in addition to TCP, is introduced. As such, a higher data throughput can be maintained because fewer retransmissions are required. As before, no channel coding is required below 0.25%, however when this rate of loss is exceeded, redundancy can be added to the stream to maintain a sufficient throughput. The need to use the local buffer therefore becomes prolonged. These added levels of coding can be applied adaptively until the associated overhead causes the data rate to exceed the maximum throughput.

In Table 3.1 the packet loss rates are provided to identify when the adaptive FEC algorithm changes the coding scheme. The area under the curves in Fig. 3.9 is when the buffer is guaranteed not to reduce, implying that continuity of the video is preserved. If the data rate or loss rate goes beyond this limit the local playback buffer will begin to empty, since the data rate is greater than the available throughput. As can be seen, the combination of TCP and FEC greatly outperforms the *TCP only* scheme in this respect.

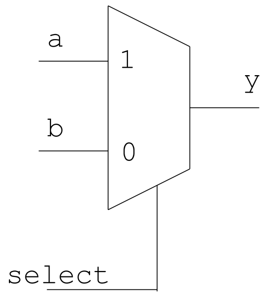
3.6 Investigation Into Hardware Implementation

In this section, a potential network solution is investigated, whereby the system described in Section 3.3 could be implemented in a hardware-based design. The premise of this system is based upon the lossy characteristics of WANs such as the Internet. Chapter 2 discusses the reasons for this and concludes that these losses are primarily due to link congestion and routing limitations. Since all hosts in this performance-enhanced network must have the ability to decode RS-encoded payloads, the network is not heterogeneous. However, since the headers of each packet will remain unchanged, routing can be achieved over a normal network, using existing infrastructure. Each border router has a hardware-based decoder which presents itself to the Operating System (OS) as a standard Ethernet interface, which ensures that upgrades made to networks are transparent. The decoders are placed at the border router of the LAN since packet losses are uncommon locally, beyond a WAN.

3.6.1 FPGA overview

A *Field-Programmable Gate Array* (FPGA) is a customisable *Integrated Circuit* (IC) which is configured and programmed with a digital electronics design. The circuit is synthesised using primitive logic blocks which are connected together to create an array of sequential or combinational logic. The process is efficient since the device is re-programmable and so design cycles are fast. Once a circuit has been designed and prototyped, it can be fabricated to a *Application Specific Integrated Circuit* (ASIC), or remain on the FPGA itself, for use in the final production system.

The desired circuit design is *programmed* using a *Hardware Description Language* (HDL), of which there are two mainstream contenders: Verilog and VHDL. Both languages differ in syntax and convention, but ultimately achieve the same objective. The language is used to describe hardware architecture that the compiler will to create, which is then synthesised into a bit stream and subsequently flashed to the FPGA. With the tools available on today's market, programmers are able to visualise the circuit they have described, before synthesis, which can be displayed right down to gate-level architecture. An example of hardware synthesis using Verilog is shown in Fig. 3.10, where an asynchronous multiplexer has been described.



```
reg y;
always @*
      if (select)
         y = a;
      else
         y = b;
```

Figure 3.10: An example of Verilog describing a hardware multiplexer.

An *Off-The-Shelf* (OTS) FPGA development kit such as that based on the Xilinx Virtex 5 includes an array of on-board peripherals such as *Serial Advanced Technology Attachment* (SATA), *Universal Serial Bus* (USB), *Video Graphics Array* (VGA) and Ethernet, in addition to the core FPGA itself. This kit provides a blank canvas with a wealth of connectivity, and can be used to prototype a wide range of embedded designs.

The advantages to using such an design are primarily operating speed and gate real-estate. The gate real-estate is the amount of gates in the array that have been used to generate the design. If the architecture is created from a very low level design, for example logic level, a highly efficient circuit can be produced. As a result of this, the speed is often very fast, especially if the encoding or decoding process is done in parallel [59].

There are however disadvantages to such a streamlined and low level design. Bespoke hardware designs take a great deal of time produce, so the initial cost, in terms of development, can be very high. Secondly the complexity of a large-scale parallel encoder, including an Ethernet stack and associated interfaces, is an incredibly time consuming endeavour to achieve from scratch. Instead, pre-made *cores* can be purchased from the FPGA manufacture [60], or other companies, which can be imported into the design as a black-box *Input/Output* (I/O) device. These modules, for example Ethernet and IP stacks, RAM or even gate-synthesised processors, can be costly to license, however.

There are a number of advantages to implementing the system in hardware: firstly it offloads the requirement for the host CPU to deal with encoding and decoding payloads, but more significantly it means that existing networks can adopt the extra level of loss protection without any changes to their present configuration. To the border router (typically

Linux based), the FPGA-customised hardware presents itself as a standard Ethernet adapter with no additional hardware requirements. Moreover, this means the system is totally OS independent.

3.6.2 NetFPGA platform

The NetFPGA [61] is a hardware accelerator consisting of a Virtex II FPGA and four Gigabit Ethernet physical layer (PHY) interfaces onboard. It is interfaced to the host PC using a *Peripheral Component Interconnect* (PCI) port. The architecture of the NetFPGA platform is shown in Fig. 3.11. Included with the NetFPGA packet are four *Media Access Control* (MAC) modules to compliment the PHY ports. The MACs themselves are licensed Verilog designs which describe the function of each device, and in turn are synthesised into hardware. Each MAC has a pair of *First-In First-Out* (FIFO) buffers for transmit and receive, again, synthesised using arrays of logic blocks.

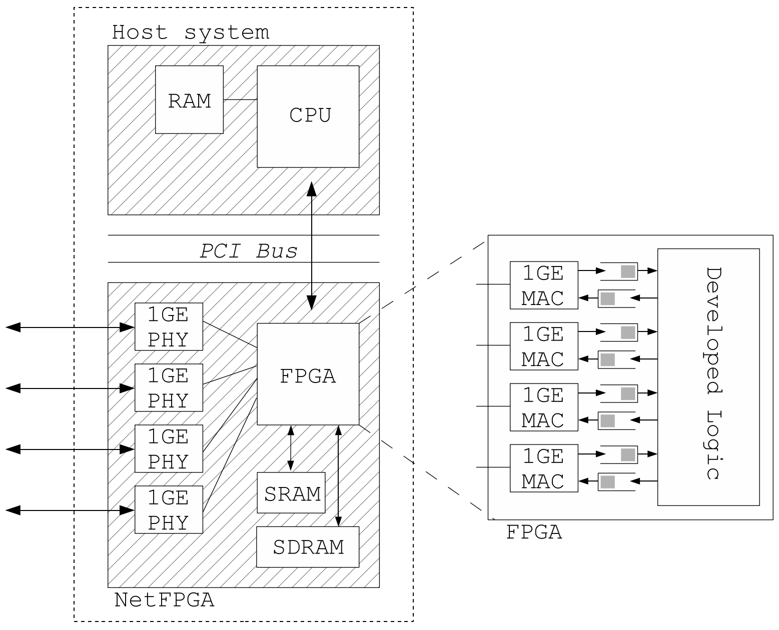


Figure 3.11: Architecture of the NetFPGA platform.

Alongside the Ethernet MACs is the self-programmed logic which controls the flow of data between the Ethernet MAC queues and the host CPU. This is the location where customised design is implemented. The host PC for the NetFPGA platform runs CentOS, which is a Linux-based operating system. The suite of tools and accessibility of the platform itself is aimed specifically at researchers to allow fast development and analysis of networking applications at the data link layer [62]. The system comes with modular references designs such as routers, switches and *Network Interface Controllers* (NICs) which allow in-depth performance monitoring of Internet systems, and prototyping of new network devices. The tools included on the host system include designers, simulators and regression tools for testing before synthesis and download.

When developing HDL to control a flow of data in a sequential manner, it is often good design practice to make use of *Finite State Machines* (FSMs). FSMs provide a structured and abstract methodology for generating sequential logic circuits, and is particularly beneficial when creating large and complex designs. They are constructed using the notion of *states* and *transitions* to alter the behaviour of the system in a sequential manner. States are transitioned between, or are persistent, on each clock cycle. Transition to another state depends on the current state and a the FSM's input signals. In addition, each state has output signals to trigger external peripherals or other FSMs.

In the next subsection a network solution is proposed which makes use of the NetFPGA platform, on-board MACs, and the FSM design principle.

3.6.3 Proposed network solution

Using the NetFPGA platform, a hardware based solution is proposed for the encoding and decoding of packet payloads using an RS FEC coding scheme. With the FEC encoder at the video source, there are two options for the deployment of the decoders. Firstly that the FEC decoders are placed in each receiving host on the *Local Area Network* (LAN) so that packets are protected from source right through to destination. There is, however, drawbacks to this technique. Each host would require the custom NetFPGA hardware accelerator for use when encoded media is delivered. However, since the majority of packet loss is found on WANs, it makes better sense to place the decoder at each border router. This solution is illustrated in Fig. 3.12.

Routing between large networks is often done using PC-based hardware and a number of Ethernet interfaces, so replacing the WAN NIC with a NetFPGA would be a transparent alteration. The NetFPGA has access to the entire packet (all layers of the IP stack) to allow decoding of the payload, which is done passively. Transparency of this operation means the

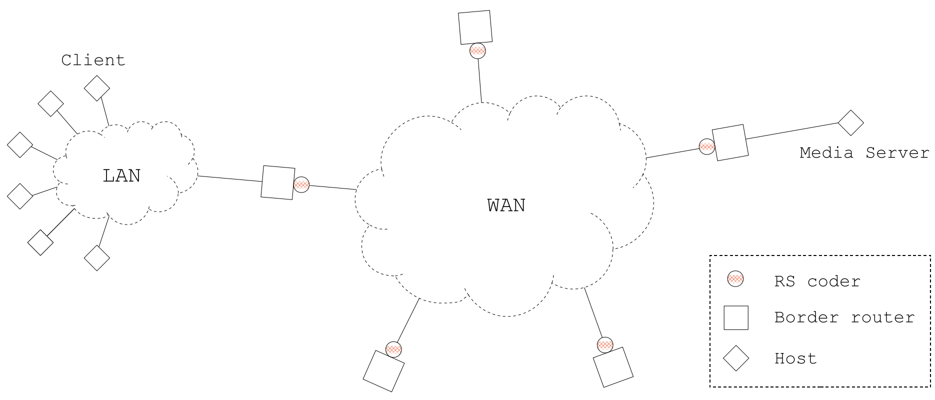


Figure 3.12: The proposed network solution using hardware-based encoders/decoders.

host router and eventual TCP end-point has no knowledge of the packet manipulation. The on-board RAM allows for packet buffering whilst *n* packets are received for each encoded block.

3.6.4 Finite State Machine design proposal

The proposed FSM is expressed using the Moore state diagram in Fig. 3.13. The top-level design consists of four states, $S0$ - $S3$, which describe the functionality of the system. Data is buffered into on-board RAM until *k* packets have been received, at which point the block is decoded and passed on to the MAC output queues.

There are three counters in the design, one to count packet length during buffering, one to count how many packets have been buffered, and one to count packet length during transmission. These are $ctr0, ctr1$ and $ctr2$, respectively.

The *machine* is initialised at state Si and immediately follows to state $S0$. In this state, packets are sequentially buffered into RAM. The data bus of the FIFO is 64 bits wide, meaning 8 bytes are saved to RAM every clock cycle. The outputs of state $S0$ are described as followed: $ctr0\_en$ enables counter $ctr0$ to count how many bytes have been buffered, and is reset every packet. $ctr1\_en$ enables counter $ctr1$ which counts at the same rate as $ctr0$, but is reset every *n* packets. Since 64 bits (8 bytes) are read on every clock cycle, $ctr0$ counts to an eighth of the MTU before transition to the next state. The RAM is designed to hold packets in sequence $1..n$, until *n* have been buffered, at which point $ctr1$ is reset and buffering begins again from the start of the RAM allocation. Data in existing RAM locations remain. This

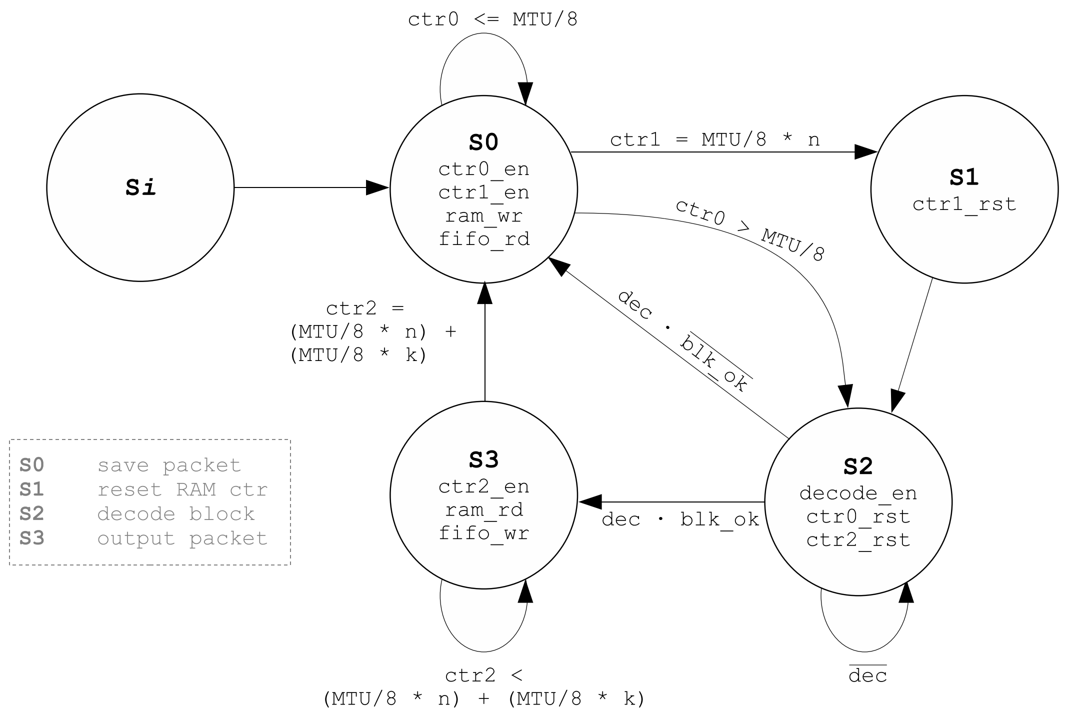


Figure 3.13: The proposed FSM for NetFPGA based decoder.

is illustrated in Fig. 3.14, and explained in detail later. Signals *fifo_rd* and *ram_wr* are asserted to allow reading and writing from the MAC's FIFO input queue to RAM.

When an entire packet has been buffered, state *S*0 transitions to state *S*2, or via state *S*1 if *n* packets have been buffered. State *S*2 asserts output signal *decode_en*, which enables the decode logic to process the contents of RAM. Signal *dec* will be asserted when the decoding process is complete. During this state, if the RAM contains enough consecutive packets to reconstruct a block, then *blk_ok* is asserted, in addition to *dec*, to signal the decode was successful. At this point, RAM positions $n+1$ .. $n+k$ will contain the decoded payload data within *k* packets, and state *S*3 will add these packets to the FIFO output queue. If the block failed to decode, state *S*2 transitions directly to *S*0 to receive another packet, before trying to decode again.

During state *S*3, outputs *ram_rd* and *fifo_wr* are asserted to control the flow of data on the bus, this time *from* RAM *to* the FIFO output queue. Counter *ctr*2 counts in much the same way as *ctr*0, in state *S*0, only this time it counts up to *k* packets.

Counters *ctr*0 and *ctr*2 are reset in state *S*2, ready for use in the following states. As an alternative in design, counter *ctr*0 may be re-used in place of *ctr*2 in state *S*3, however a fourth state would be required to reset the counter before transitioning back to state $S0$. It is a trade-off between an increased number of logic gates (due to an additional counter) and a further clock cycle (due to another state).

	A	B	C	D	E	F	G	
a	**1**	-	-	-	-	-	-	✗
b	1	**2**	-	-	-	-	-	✗
e	1	2	3	4	**5**	-	-	D
h	**8**	2	3	4	5	6	7	✗
i	8	**9**	3	4	5	6	7	✗
j	8	9	**10**	4	5	6	7	✗
k	8	9	10	**12**	5	6	7	✗
l	8	9	10	12	**13**	6	7	D
m	8	9	10	12	13	**14**	7	✗
p	16	**18**	10	12	13	14	15	✗
q	16	18	**19**	12	13	14	15	D
r	16	18	19	**20**	13	14	15	✗

Figure 3.14: Proposed utilisation of RAM for machine cycles $a - r$

Fig. 3.14 illustrates the proposed utilisation of the on-board RAM for each iteration of state $S0$, and the functionality of the decoder state, $S2$. In this example an RS code with $(7,5)(n,k)$ parameters is used. The n RAM locations are labelled A to G, and each write cycle a to r. The packet sequence number is shown in each RAM location, and emboldened to show which packet was written for that cycle.

When state $S0$ has completed buffering the first packet to RAM (FSM cycle a), only one of n locations are occupied: location A with packet 1. Counter $ctr1$, which keeps track of which RAM location should be written to next, has not yet overflowed and so the state transitions to $S2$ for decoding. At this point only one packet is in RAM, and so $S2$ transitions back to $S0$. The red cross in Fig. 3.14 denotes a decode failure at FSM cycle a, which is expected at this point. This remains the case until cycle e when the RAM contains k packets, at which point decoding is a success. The packets used during the successful decode are shown boxed.

At cycle h, all RAM locations are full, and so packet 8 is written to location A in accordance to $ctr1$, which has been reset in state $S1$. The data in locations B to G remain unchanged. Packets are added until the next complete block can be decoded, at cycle l. Notice however, at cycle k, packet 11 has been lost during transmission, but the subsequent decoding is still possible. Similarly at cycle p, packet 17 has been lost, however the block of packets (starting from 14) need only one more packet for a successful decode. This is satisfied at cycle q.

It is assumed all packets are equal length, with padding appended at the encoder as necessary. The size of the RAM locations can therefore be fixed.

Packets arriving in an unordered sequence will fail to decode if the ordering is out by more than n packets, since block overlap will occur. A larger RAM allocation will allow for greater unordering, but will add to the delay caused by the buffering process.

The design does not cater for adaptive coding at this stage, since the overflow value of counter $ctr1$ is fixed. The FEC decoding logic can be designed or licensed as a completely separate module for clear abstraction from the main design in Fig. 3.13.

3.7 Conclusion

It has been shown how passing data through a coding scheme to protect against loss can improve the throughput of a video stream in comparison to simply allowing TCP to retransmit any lost packets. Recovering any Erasures is far more efficient than requesting the data be sent again, but this must be balanced with the amount of overhead which accompanies such a procedure. Care must be taken when choosing the coding levels to adapt between. The results show how much benefit in throughput one can expect, and whether the additional overhead makes it feasible.

The most significant result is the use of adaptive FEC in a real world test environment achieving an increase in available throughput, compared to using TCP alone. The case study shows how this can be applied practically and, as a result, significantly improve the quality of service experienced by the user, whilst accounting for any limitations.

If the *User Datagram Protocol* (UDP) were to be used, or the selective retransmission adaptation of TCP [63], in an attempt to further improve performance, the playback of video would become distorted as a result of loss. This is not something users are nowadays accepting of, especially with the impeccable quality of popular playback systems such as the BBC iPlayer [13]. It is also worth noting that when using UDP, the congestion control algorithms do not apply, so a large influx of datagrams may cause excessive network congestion.

A proposed network solution was investigated, using a hardware-based FPGA design based on the NetFPGA platform. A *Finite State Machine* (FSM) was produced to describe the functionality of the design, and is the basis for any future work in this research. The FSM can be used to create the necessary HDL code to generate a top-level sequential logic circuit which will drive a complete system.

Chapter 4

Hybrid TCP/UDP video transport

In this chapter, a hybrid TCP/UDP transport is proposed, specifically for H.264 encoded video, as a compromise between the delay-prone TCP and the loss-prone UDP. When implementing the hybrid approach, it is argued that the playback quality at the receiver often need not be 100% perfect, providing a certain level is assured. Reliable TCP is used to transmit and guarantee delivery of the most important packets, providing a level of unequal loss protection. This allows use of additional features in the H.264 standard which simultaneously provide an enhanced playback quality, in addition to a reduction in throughput. These benefits are demonstrated through experimental results using a test bed to emulate the hybrid proposal.

The proposed system is compared with other methods of protection, namely *Forward Error Correction* (FEC), and in one case show that for the same bandwidth overhead, FEC is unable to match the performance of the hybrid system in terms of playback quality. Furthermore, the delay associated with such an approach is measured, and examined for its potential use as an alternative to the conventional methods of transporting video by either TCP or UDP alone.

4.1 Introduction

There are two conventional methods for transporting video over IP based networks: the first is using connectionless UDP based transport where datagrams are transmitted without any request or acknowledgement from the receiver. If packets are lost in transit, there are no retransmissions or attempts to recover the lost data. This method has the advantage of ensuring low end-to-end transmission delay, but at the expense of the inability to guarantee data delivery. Moreover, if a router within the transmission path is dropping packets due to congestion, there is no method to signal the transmitter to reduce the transmission rate [3].

When using TCP as a transport method, the delivery of each packet is always guaranteed, since lost packets will be retransmitted when a break in packet sequence is detected. This will, however, cause additional delay which may cause playback of video to become interrupted [64].

With video transmission, it is often a non-critical requirement that all the data is received perfectly, providing the important keyframes and header information is received, some loss of lesser-important data yields only minor imperfections in playback [8]. In the new H.264 video standard, it is possible to organise the importance of the data into partitions during the encoding process [23], which allows greater protection for partitions with higher importance. This is discussed further in Section 4.2.

Packet loss in *Wide Area Networks* (WANs) such as the Internet is recognised to be predominately caused by router congestion along the connection path [53]. These losses usually manifest themselves not in a unique fashion, but as a series of consecutive losses known as burst loss. Burst lost can be especially detrimental to a video stream because there is greater probability that the losses will cause critical information, such as Keyframe and *Network Abstraction Layer Unit* (NALU) header information, to be lost [65]. This can cause very detrimental results to the video playback, and in severe cases, prevent the video from decoding.

This research looks at the benefits of combining TCP and UDP to provide a non-perfect, but more robust, method of delivering video content in lossy networks. The work is presented as a hybrid TCP/UDP transport for video content delivery.

In Section 4.2, the H.264 standard is introduced and the ability to partition video, with aim to protecting the most important parts, is discussed. Section 4.3 details the rationale behind the hybrid transmission scheme. Section 4.4 explains the configuration and set up of the experiments conducted, and the results are followed in Section 4.5. Finally, conclusions are drawn from this research in Section 4.6.

4.2 Background and previous work

H.264 video has a frame structure which can be categorised in to several classes. Each frame of video is grouped together to form a *Group Of Pictures* (GOP). The GOP consists of an initial *Instantaneous Decoder Refresh* (IDR), or I-frame, and then a number of *predicted frames* (P-frames) thereafter. The I-frame contains all the data required to build a complete picture and does not depend upon any other frames within the GOP. They are said to be temporally independent, and are typically the largest type of frame in terms of encoded bits. A predicted frame contains only part of the picture along with a series of motion vectors which reference data in previous frames within the GOP. Since the content of consecutive frames typically change very little, these references can be used build a frame with only the updated areas requiring new information. This allows significant compression of a GOP [23].

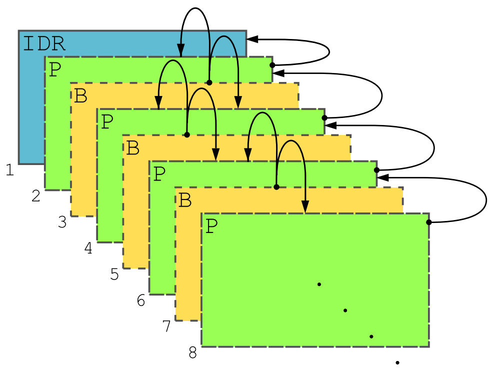


Figure 4.1: Make up of a typical H.264 GOP

In addition, for the case of newer codecs such as H.264, there are *Bi-directional Predicted frames* (B-frames) which allow for predictions based on previous and future I- or P-frames. In usual cases, predictions are not made from B-frames [8]. Fig. 4.1 illustrates the structure of a GOP, in particular how the I-frame does not depend upon any other frame, and the typical source for different predicted frames. Fig. 4.2 illustrates how errors in one frame can lead to frame degradation throughout much of the remaining GOP. In this case, frame 4 has sustained damage due to packet loss which, the adjacent frames, 3 and 5, depend upon. The predictions made are based on corrupted data and the errors are propagated. The corruption then follows into frame 6 and beyond; the remaining frames in the GOP are all affected. The frequency of I-frames can be increased to minimise the duration of corrupt video however will require additional bandwidth.

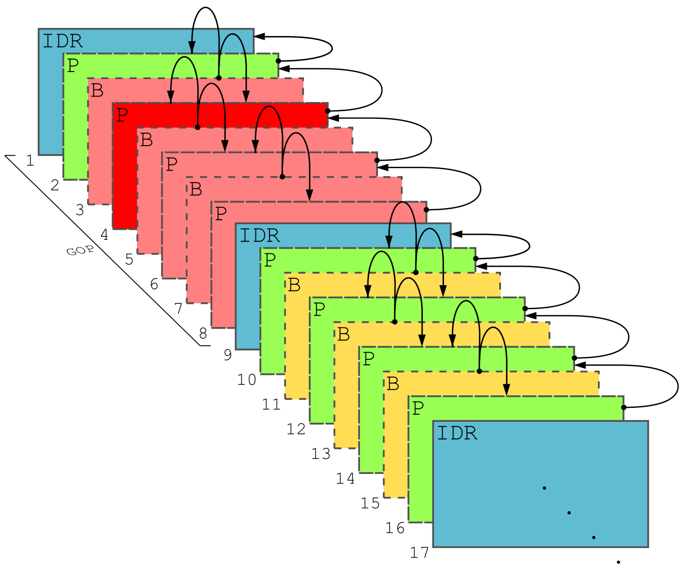
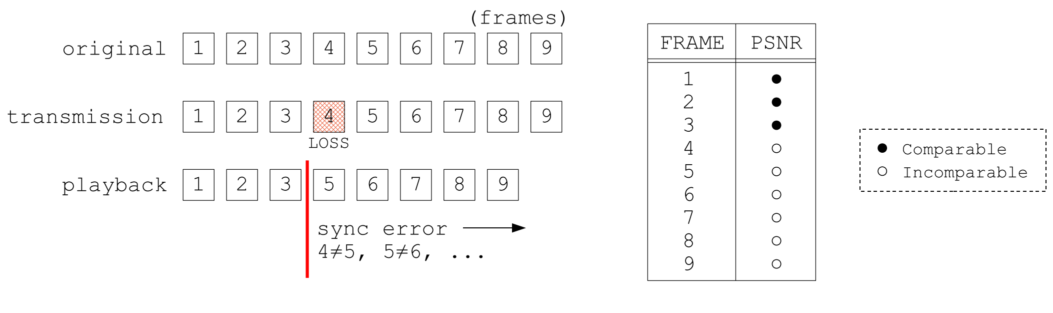


Figure 4.2: Progressive GOP degradation

The level of picture degradation which the GOP sustains is compared to the source video using the *Peak Signal-to-Noise Ratio* (PSNR), which was detailed in Chapter 2. When comparing two images of identical content with a similar quality, one can be considered a noisy approximation of the other. It important to note that when using PSNR to compare two similar videos, the comparative frames must remain in synchronisation throughout to ensure only the noise is considered. If a frame is dropped by the decoder, due too much data loss, the resulting video will become slightly shorter in duration than the source. This caveat makes the videos incomparable in full, and is illustrated in Fig. 4.3.



FRAME	PSNR
1	●
2	●
3	●
4	○
5	○
6	○
7	○
8	○
9	○

Figure 4.3: PSNR comparisons are meaningless if a video becomes desynchronised with the source due to a dropped frame.

In this example, a series of video frames are transmitted, and then the PSNR is calculated in comparison to the original sequence. Sufficient packet loss occurs during transmission causing frame 4 to be dropped during playback. When comparing the PSNR on a per-frame basis, the sequence of pictures is now one less, and so comparisons between frames become unsynchronised, leading to noise, as well as content, being compared for differences. The quality measurement becomes meaningless. During each of the experiments, which will be detailed in Section 4.4, it is ensured that the decoded videos match that of the source in terms of frame count.

Data partitioning is a new feature provided by the H.264 standard which provides further resilience to degraded playback quality by ordering the data into three partitions. This provides a number of advantages, but principally to allow prioritisation of important parts of the video stream. The most critical partition, partition A, contains units such as slice headers, quantisation parameters and motion vectors. The other two partitions, labelled B and C, contain only residual information of intra-coded and inter-coded macroblocks, and are of lesser importance to the decoding process [8]. Using this method, the data in different partitions are assigned a NALU of their own.

With this mechanism in place, protection methods such FEC can be applied to protect only the most important parts of the video stream, or adaptively in a hierarchical fashion to allow unequal loss protection [66] [67].

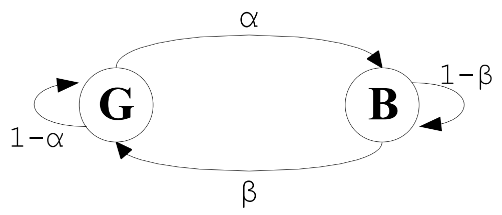


Figure 4.4: Gilbert's 2-state model

It was introduced in Section 4.1 how losses in WANs are predominately burst losses due to congestion. Implementation of packet loss in the system will be based on Gilbert's two-state model [68] to more accurately simulate the burst characteristics of a typical lossy Internet path.

Fig. 4.4 illustrates the 2-state model. It consists of a good state, *G*, and a bad state, *B*. When in the good state packets are received by the destination; in the bad state the packets are lost. The probability of transition from the good state to the bad state is given by α, and from the bad state to the good state by β. It is capable of modelling uniform random loss and burst losses, the latter of which implies that state B is persistent as time t progresses.

4.3 Hybrid TCP/UDP

Previous research has shown that packet loss can have a severe consequence on throughput when utilising TCP because of the congestion control algorithms used, and this in turn can cause video playback to start and stop significantly [64]. Attempts to reduce the number of retransmissions required and maintain a stable throughput can be aided by combining FEC with TCP [10], which was presented in Chapter 3. The combination TCP and FEC guarantees delivery of the video content with perfect quality, although at a reduced throughput when packet loss is experienced. Conversely, if one is to use solely UDP, there is no additional delay inherent with TCP re-transmissions, however packet loss is uncontrolled and can cause significant picture degradation. In Section 4.1 it was stated that it is often the case where video playback need not be perfect, providing the picture quality maintains an acceptable level. The solution presented is to combine TCP and UDP to provide a hybrid transport approach - permitting packet loss for lesser important video partitions, whilst protecting partition A, the most important.

This provides two main benefits: firstly that the TCP congestion control algorithm is not over used, which has been shown to cause huge delays in lossy network conditions [64], and secondly, that lesser important data which can be lost with minimal impact to video quality is not retransmitted unnecessarily, saving bandwidth and helping towards overall channel conditions. The saving in bandwidth and lesser reliance on TCP allows the continuity of video playback to be better preserved in exchange for slight degradation in the quality of the playback. In addition, because partition A data is able to be guaranteed, more advanced features in the H.264 standard can be implemented which would otherwise be inadvisable in error prone networks. In the results are included the use of B-frames which other research has not considered [23] [67] or discussed [69]. It is shown that when protecting partition A, use of B-frames provide significant benefits such as better video quality and compression rates.

4.4 Experiment Setup

A YCrCb sourced video clip is encoded to H.264 using the *JM* reference encoder. The extended encoder profile is implemented to allow partitioned slices: A, B and C. The encoder allows for packetised output whereby the encoded video stream is encapsulated into RTP packets, and saved to file. Two videos are encoded from the source video clip, one with a *IPPPPP...* GOP structure and a second with a *IPbPbP...* structure, which is illustrated in Fig. 4.5.

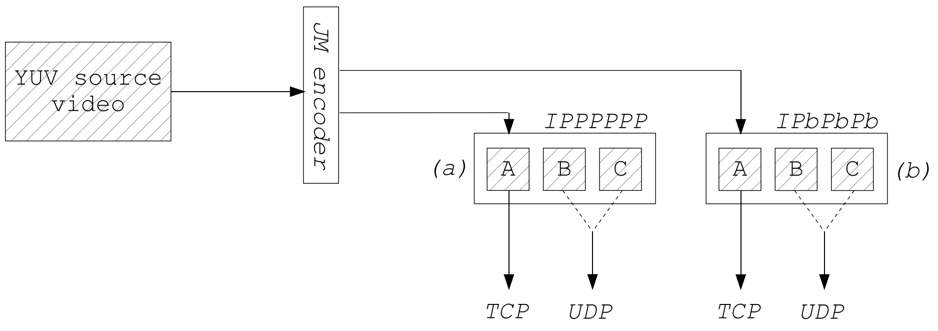


Figure 4.5: The JM encoder and packetiser for both cases, with (b), and without (a), B-frames.

They differ in that the first contains only an initial I-frame followed by P-frames, and the second an initial I-frame followed by an alternating sequence of P- and B-frames. By sequentially parsing each file the NALU type can be read, and hence determine which partition the packet has been assigned to. Using a modified version of *rtp_loss*, a utility included with recent versions of *JM*, it is possible to protect specific partitions in order to examine the most robust combination of packet loss protection schemes.

The primary objective is to demonstrate the benefits of protecting the most important partition, partition A, with reliable TCP transport, whilst allowing all other partitions to be sent using UDP. The hybrid approach is implemented as described in Section 4.3 and the results compared with protection methods such as FEC.

With a semi-corrupt RTP stream, produced by the modified *rtp_loss* utility, decoding of the video stream is executed and the playback quality is analysed. The original YCrCb video is used as a reference; the quality between the two encoded videos (one with B-frames and one without) is ensured to match closely to allow comparison. The decoder is configured to use the *temporal* algorithm for error concealment.

The quality of playback in comparison to the source video is measured using PSNR. The average PSNR is taken over the entire video playback. Only the luminance channel is considered, which holds the majority of the picture detail.

The work presented in [66] and [8] demonstrate the benefits of FEC when used to protect video streams, but only when using UDP as a transport method. The work presented in this Chapter uses TCP with FEC as an alternative approach, but UDP results are included for comparison to this existing work.

When implementing FEC, coding parameters (n,k), as $(7,5)$ and $(7,6)$ are used to allow recovery of up to 2 Erasures per block of 7 packets, and 1 per block of 7, respectively. The latter is effectively a parity check, and very simple to implement. The choice for these parameters is discussed in the analysis of the results. The clip used was the *paris* test sequence, which consists of 1065 frames in CIF resolution, 352x288.

α	β
0.0004	0.9792
0.0109	0.7915
0.0141	0.8268
0.0178	0.8548
0.0218	0.8392
0.0347	0.9091

Table 4.1: Loss probability parameters for Gilbert's 2-state model [1]

Our burst loss model is based on the 2-state markov model with parameters from [1], which are shown in Table 4.1. These parameters give a range of average packet loss rates between 0% and 3.5%, which is typical of the loss rates found on the Internet [1] [70].

To differentiate between different partitions, the NALU type must be known for each encoded packet, allowing different protection schemes to be used. Listing 4.1 provides an exert of the code used to achieve this within the *rtp_loss* program.

Listing 4.1: Detecting partition A

```
//detect NAL unit type
if ((buf[12] & 0x1F) == 2) {
    //NALU is in partition A
} else {
    //NALU is in partition B or C
}
```

The code is written in C. The *buf* variable contains the packet data which, at the lower 5 bits of the 13th byte, contains the NALU type [71]. Performing a logical *AND* operation on this byte with a value of *0x1F (b'00011111)* masks off the lower 5 bits. A check for equality with *2* can then be made to deduce partition A [71]. The structure of the NAL header is shown in Fig. 4.6.

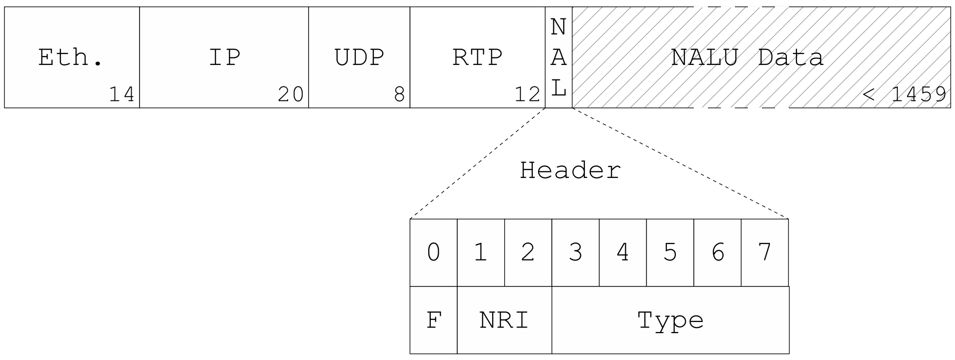


Figure 4.6: The header of a NAL unit, containing the partition type.

In the tests, each packet contained a full NAL unit, therefore no fragmentation or packing amongst packets occurred.

In addition, to calculate the delay associated with using a TCP-based hybrid scheme, the video stream is transmitted over a real network and the Linux utility, *Traffic Control* (tc), was used on a bridge to drop packets in the exact positions as the simulated case. Using the new version 2 of *Network Emulator* (NetEM), it is possible to produce burst losses described directly from a loss pattern file, as generated by the simulator and parameters in Table 4.1. The packets are captured at the destination with *tcpdump* and the *Round Trip Time* (RTT) of each packet is analysed. The test bed system architecture is shown in Fig. 4.7.

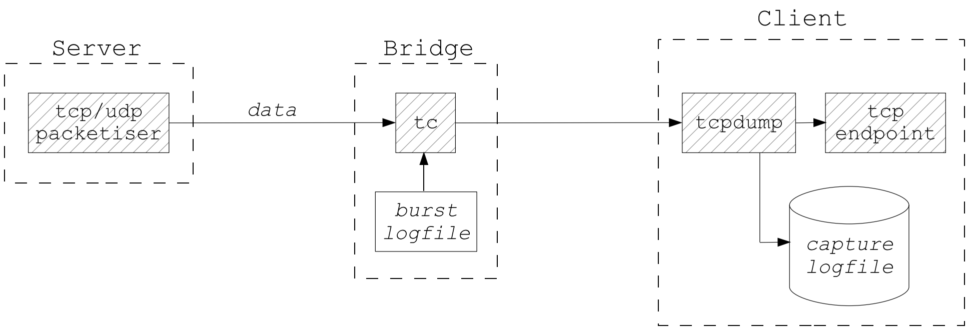


Figure 4.7: The test bed system architecture.

4.5 Results

Encoding the video with B-frames reduced the encoded video stream size by a significant 18.6%, compared to when encoded with only P-frames. The initial quality between the two videos is similar, with only a 0.07dB difference. In addition, a 3.5% increase is noticed in the use of partition A when B-frames are used, but only when partition A is guaranteed. The video stream is less robust to packet loss when no form of protection is implemented.

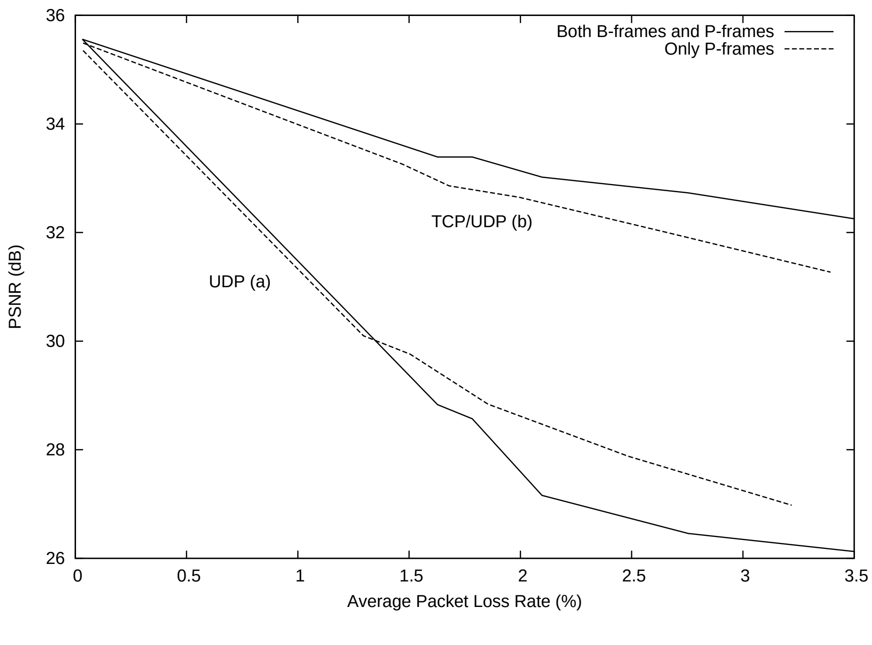


Figure 4.8: Comparison between the use of B-frames *(IPbPbPb...)* and P-frames only *(IPPPPPP...)* when protecting partition A with TCP, and with no protection (UDP only)

In Fig. 4.8 an unprotected video stream, labeled (a), shows a lesser quality when B-frames are used, of 1.04dB in PSNR, on average, when compared to the use of P-frames only. If the protection of partition A can be guaranteed using reliable TCP (labeled (b)), an expected increase in video quality is noticed. More significantly, however, the use of B-frames in this case conversely offers *better* quality than when using P-frames alone, and yields a PSNR increase on average of 1.04dB in comparison. From this, it is concluded that the use of B-frames offers two main benefits - greater compression and better playback quality. The use of B-frames when streaming in loss prone networks is made viable if used in conjunction with the hybrid transport approach, and without the necessity to rely completely upon TCP.

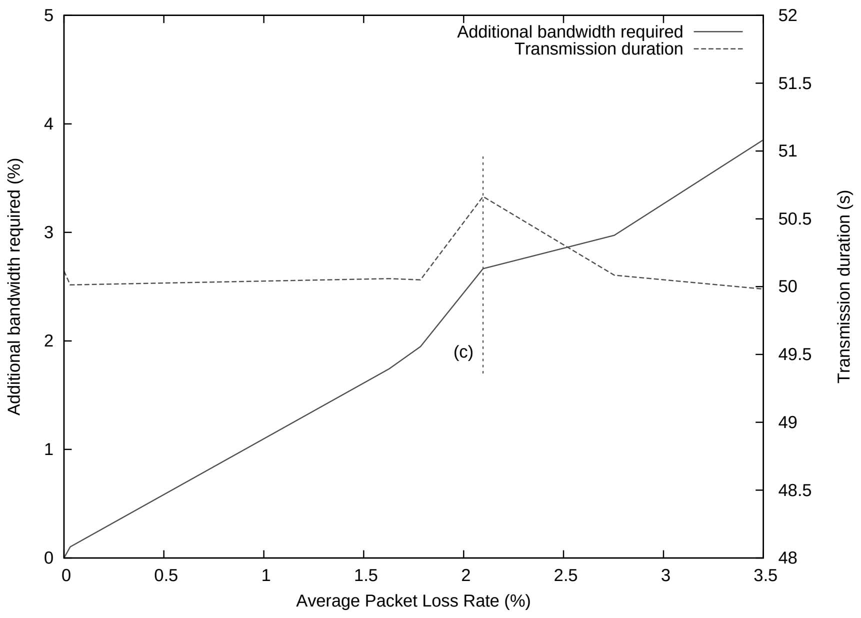


Figure 4.9: Additional bandwidth required for TCP retransmission, and duration of transmission for each experiment

Fig. 4.9 shows the additional bandwidth required to retransmit lost packets at the different packet loss rates applied throughout the tests. At its maximum of 3.5% average loss rate, a 4% increase in bandwidth is required, the additional 0.5% is accounted for by multiple re-transmissions. In the same figure, the duration of transmission for the TCP part of the system is included, and it is noticed that on the most part, delay is not significant even at the upper average loss rates. The biggest deviation seen is just under a second, at $0.72s$, which coincides with a sudden increase in retransmissions. This point is labelled in Fig. 4.9 at (c). Importantly, this increased delay does not affect playback quality in the hybrid system, since all losses are retransmitted. Moreover, the delay is not significant enough to cause any interruption to playback, nor to prompt re-buffering.

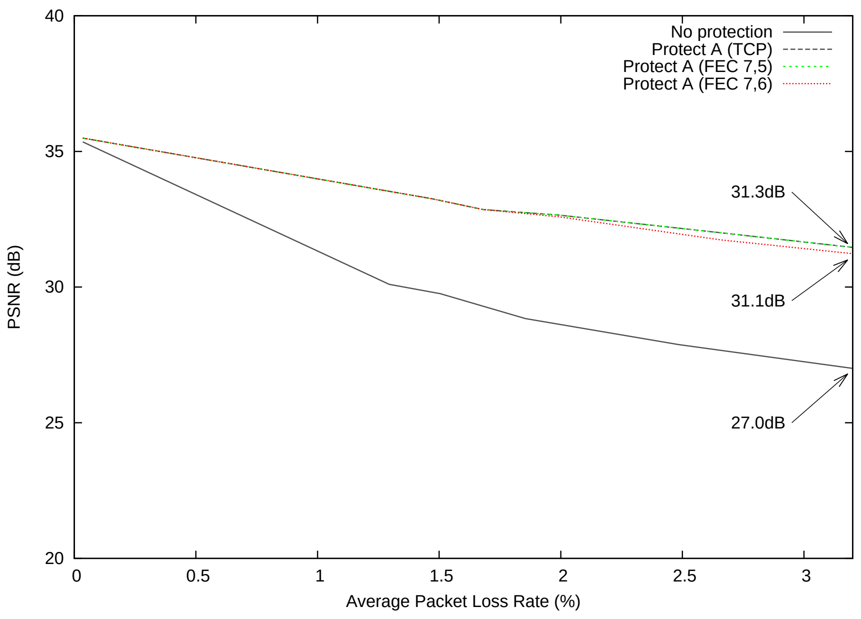


Figure 4.10: Protecting partition A with FEC, without use of B-frames.

In Figs. 4.10 and 4.11, the results are compared with two FEC protection schemes. Firstly, an (n,k) coding scheme is considered with parameters $(7,5)$. Up to two losses can be recovered per block of 7 packets. From the experimental results the coding scheme is able to correct all losses throughout the range of packet loss rates. The quality of the decoded video is the same as that from the TCP protected partition A, for both source videos, when using B-frames (Fig. 4.11) and then only P-frames (Fig. 4.10). In this case however, a constant 9.89% requirement for additional bandwidth is calculated, even at low levels of loss, to cover the excess redundancy transmitted.

The second protection scheme uses parameters $(7,6)$ and is chosen for the calculated overhead of 4% (when used to only protect partition A), matching closely to that of the hybrid TCP/UDP scheme. It is seen, however, that for the same additional bandwidth required to protect partition A, the FEC scheme is not able to recover all the losses experienced beyond 2% packet loss rate, leading to a reduction in video quality, particularly when B-frames are used (Fig. 4.11). This demonstrates the advantage to the hybrid approach given the same bandwidth requirements.

The low transmission delay, even at the upper loss rates are apparent because the majority

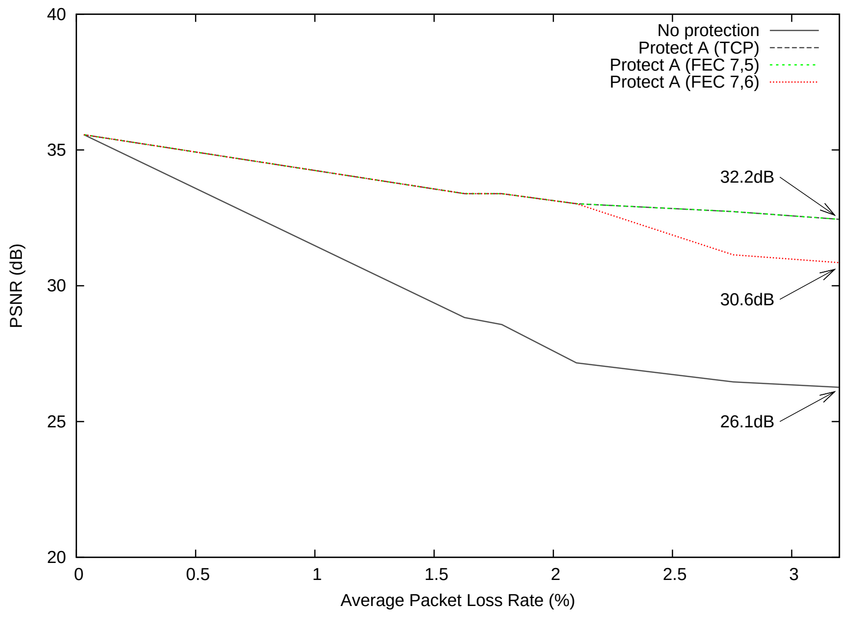


Figure 4.11: Protecting partition A with FEC, and use of B-frames.

of the video stream is being transmitted using UDP, and TCP only has to control a limited amount of packet losses. As such, the TCP window size remains relatively large and delay remains minimal. Since the combination of both TCP and UDP is being used, it poses the potential for UDP packets to arrive at the destination before they are required, ie, before more important partition A data. The issue will mostly be alleviated with a buffer at the client side, and it has been shown that the size of this buffer need only be greater than the maximum RTT experienced, which in the tests is found to be minimal. If the delay were to grow beyond the length of the client buffer, playback would simply be required to pause until the buffer becomes re-saturated. Since each packet is being sent from the same process, in a round-robin fashion, the UDP stream would not continue to send if the server is waiting for a TCP acknowledgement. Any packet reordering is resolved by the use of RTP.

4.6 Conclusion

A hybrid TCP/UDP approach is presented for transporting H.264 encoded video in error prone networks, and a number of benefits to the proposed system are shown. The system is presented as a compromise between delay-prone TCP and loss-prone UDP, and it is argued that the quality at the receiver often need not be 100% perfect, providing a certain level is assured.

The advantage to using B-frames is demonstrated when encoding an H.264 video stream, and is shown thorough experimental results to yield a benefit in quality of 1.04dB on average, and a 18.6% saving in bandwidth when compared to an encoded video which uses P-frames alone. The quality difference before transmission of the two videos were within 0.07dB. The use of B-frames is made viable using the hybrid approach, because partition A is assured, without the necessity to rely completely upon TCP.

Furthermore, it is shown that the hybrid system does not cause significant levels of delay or jitter, and it is shown experimentally that this holds even at moderate levels of packet loss.

The proposed system is compared with other methods of protection, namely FEC. It is shown in one case that for the same bandwidth overhead, FEC is unable to match the performance of the hybrid system in terms of playback quality.

Chapter 5

Pause Intensity

Video nowadays is rarely transmitted over the Internet using UDP - since the birth of YouTube [12], and later, BBC iPlayer [13], TCP has been the choice for video transport now that the bandwidth of domestic broadband is often in excess of the source video data rate. Witnessing picture degradation is unusual, since the playback will not attempt to continue if data is missing; playback simply pauses until the local buffer is re-saturated. As such, when using purely TCP, quality measurement using PSNR is meaningless, and so a new method of assessment needs to be realised.

This chapter investigates a new objective measurement technique for assessing video playback quality for the services delivered using TCP as a transport layer protocol. A new metric is defined as *pause intensity* which characterises the quality of playback in terms of its continuity. In the TCP environment, data packets are protected from losses but not from delays. Using packet traces generated from a real TCP connection in a lossy environment, playback of a video can be simulated, and buffer behaviours monitored, in order to calculate pause intensity values. Subjective tests are also conducted to verify the effectiveness of the metric introduced and show that the results of pause intensity and the subjective scores made over the same real video clips are closely correlated.

5.1 Introduction

In 1998, the development of the *Real-time Streaming Protocol* (RTSP) revolutionised the way multimedia content was delivered over the internet and introduced the ability to stream easily, modest-quality video to home users. The video itself was streamed predominately using the *Real-time Transport Protocol* (RTP) which had been developed only two years previously. RTSP allowed the end user to control the streamed video's playback, providing options such as play, pause and seek. The RTSP control messages were sent back to the streaming server which, in turn, changed the RTP video stream appropriately. When the video stream was live, these controls were obviously limited. There were other proprietary streaming protocols such as the *Real Data Transport* (RDT) protocol used by RealNetworks in their popular RealPlayer client, but the functionality was much the same as RTP.

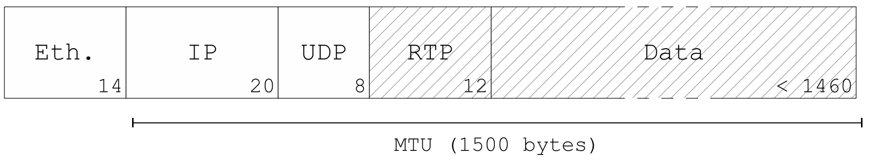


Figure 5.1: Structure of a typical RTP packet [11]. Headers can vary in size due to optional attributes.

RTP uses the *User Datagram Protocol* (UDP) as its underlying transport mechanism, with an additional set of headers as shown in Fig. 5.1. Since UDP cannot control errors or rely on re-transmissions, packet loss was expected within congested networks, and so the video codec had to allow for minor loss of data without greatly disturbing the playback quality. A buffer at the client allowed for jitter (variation of inter-packet arrival times), and also for the video data rate to be equal or marginally higher than the client's connection speed; of course at the expense of having to wait for a short period before playback would begin. For live video streams, the data rate had to be less than the maximum available throughput to ensure the client's buffer remained full. However, packet loss meant that the quality of playback would almost certainly be degraded, regardless of buffer size.

Modern internet video playback applications are predominantly Flash based [72], such as Youtube [12] and, in the UK, BBC iPlayer [13], but the more conventional standalone media players such as Windows Media Player and *Video Lan Client* (VLC) [73] are still widely in use and under constant development. The advantage with Flash applications is that content can be seamlessly embedded into a website to create a more integrated feel, and are often specifically tailored to the content they are delivering. The simple user interface has made it popular amongst developers to adopt such methods, as opposed to using a separate application for playback.

In either case, TCP is now the preferred method of transport rather than a RTP/UDP combination for the following reasons: it is recognised that the majority of video content delivery over the internet is not live, the user's bandwidth is often many times greater than the video data rate, retransmissions allow for recovery of lost packets, and streams remain TCP-friendly [50], which is important for scalability. Being able to guarantee the delivery of a video packet also allows the use of newer codecs, such as H.264, with high compression rates, which are less resilient to data loss [74].

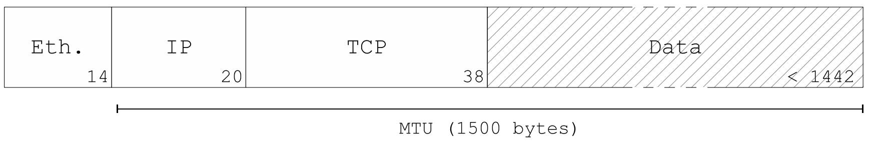


Figure 5.2: Structure of a typical TCP packet [4]. Headers can vary in size due to optional attributes.

TCP-friendly traffic is defined as such when a serving end-point reduces its throughput when a loss occurs within the channel. This has become more important as the Internet has grown [15] and is one of TCP's main advantages. The structure of a typical TCP packet is shown in Fig. 5.2. Since UDP provides no mechanism for feedback, any congestion caused by an influx of packets cannot be controlled so the link becomes, and remains, saturated which results in packet loss.

Buffering allows time for the retransmission of lost packets to occur without the associated delay interrupting the playback, since the loss is recovered before its playback. It follows, therefore, that playback issues due to jitter are almost completely alleviated, since the inter-arrival time of packets are often only a few milliseconds.

Video playback buffers operates on a *First In, First Out* (FIFO) principle, whereby arriving data is sequentially buffered and leaves in the same order it arrived. However, since reordering of packets can occur on packet-switched networks, the opportunity for re-sequencing the data at this point is viable (Fig. 5.3). The generalised term *buffer*, can be either a packet based buffer, video bitstream buffer, or both. Re-sequencing of the data is done before the payload is extracted from the packet since the sequencing information is contained within the packet headers.

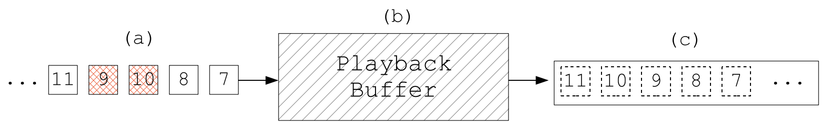


Figure 5.3: Unordered packets caused by jitter (a) can be rectified (c) during the playback buffering stage (b).

It was shown in Chapter 3 how adding *Forward Error Correction* (FEC) to blocks of packets can significantly decrease the number of retransmissions required when packet loss occurs [75], which means that the buffer is less likely to become exhausted. There is, however, still a point where the packet loss is sufficiently high to cause the playback buffer to empty, and the playback to momentarily pause. When this occurs it is very difficult to measure, objectively, the effect this has on the perceived quality of the video stream.

Existing objective measures of perceived video quality, such as the *Peak Signal-to-Noise Ratio* (PSNR), are more suited to objectifying the performance of compression algorithms [76]. In addition, this type of measurement is frequently used when dealing with a UDP video stream [77], since lost packets are not retransmitted, leading to degraded playback in comparison to the original. As such, this method of quality assessment was used in Chapter 4.

When using TCP, however, packets are always recovered if loss occurs, and so quality degradation is rarely seen. This indicates the unsuitability of PSNR as a metric for quality assessment in this case, and alternative metrics for this type of applications are not yet available [78]. For this reason, the focus is turned towards buffer behaviour as video pausing due to buffer under-runs has become the main source of quality concerns in video steaming using TCP.

The research in this chapter proposes a novel method to characterise the perceived quality of a video stream by measuring the degree of interruption caused by pauses during playback. It is shown how packet loss affects the number of pauses, and the average duration per pause, for a given set of playback criteria such as the playback rate, available bandwidth and buffer size. The pause intensity metric is then proposed for measuring the effect on video playback, which is validated by showing its correlation with the subjective measurements conducted in the same streaming environment.

Section 5.2 will outline some preliminary background research which has been considered during this work, covering Traffic Theory and how this can be applied to this work. It also sets the metric for why this research has been carried out. Sections 5.3 and 5.4 explain how the testbed and simulator results were collected and is followed by the results in Section 5.5. Results from a subjective video quality survey are presented in Section 5.6 to validate the model. Finally, conclusions are drawn in Section 5.7.

5.2 Pause Intensity

Traffic Theory has always been of critical importance when designing telecommunications networks, as it provides the system bounds which ensure efficient operation [79]. Originally, it was used in analogue switched telephony, where a probabilistic model can be derived to predict the unfavourable situation of call-blocking. This occurs when there were more calls being placed than outgoing lines available and ultimately led to a loss of service for anyone attempting to place a call. The traffic intensity is an objective measurement for modelling the performance of a resource-limited telephone network given the number of simultaneous calls per unit time [80]. With only two parameters - the call arrival rate and the average call duration - the traffic intensity can be determined, and is represented by the unitless Erlang loss formula [79]. With the recent switch to digital telephone networks, traffic intensity adapts nicely to suit the new technology and, instead of outgoing lines, available throughput is used along with the incoming data rate and packet size. Despite the similarities between analogue and digital switched telephone networking in terms of the Erlang loss formula, there are very few other internet-based applications which utilise this method of modelling [81].

The traffic intensity model suits the requirements for measuring the quality of a video in terms of network resources perfectly. The premise is adapted to measure the intensity of *pauses* during playback. Instead of measuring the performance of the video stream and predicting how this may affect playback, the playback analysed directly and its quality described in terms of breaks in its continuity.

In a TCP network, the service delivery time varies as lost packets are retransmitted, causing delays and variable incoming data rates at the receiver buffer. If the playback rate,

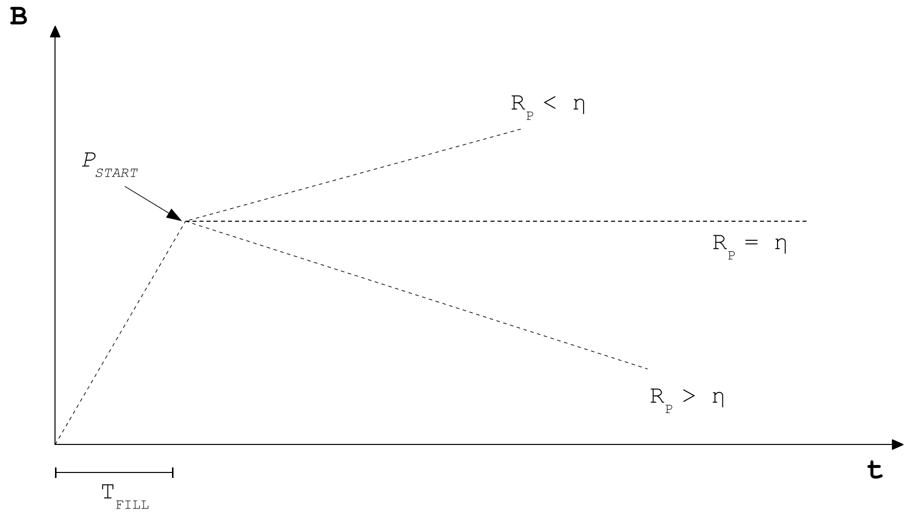


Figure 5.4: Buffer trend with respect to available throughput

R_P, is greater than the incoming data rate (or throughput, η) the buffer will be utilised and eventually become exhausted, causing the playback to pause. Fig. 5.4 shows, in addition to this case, when the throughput is greater than the playback rate and when both the throughput and playback rate are equal.

The time between $t = 0$ until P_{START} illustrates the pre-buffering period, T_{FILL}, during which playback does not commence. When the buffer is sufficiently full, playback begins. Fig. 5.5 further examines the general case where $R_P > \eta$. In this example, the duration of playback, T_P, is consistently interrupted for duration τ.
Whilst playback is paused, the buffer fills at a rate of η. The period of the play/pause cycle, T_C is expressed by $T_P + \tau$, giving a frequency $f = \frac{1}{T_C}$. It follows, therefore, that the duty cycle is:

$$D = \frac{T_P}{T_C} \tag{5.1}$$

To measure the effect of buffer exhaustion on the perceived playback quality, a metric termed pause intensity is introduced, which is contributed jointly by both the number and the duration of pauses during playback. Given the number of pauses per unit time u and the average duration per pause v, the pause intensity I_p is defined as

$$I_p = uv \tag{5.2}$$

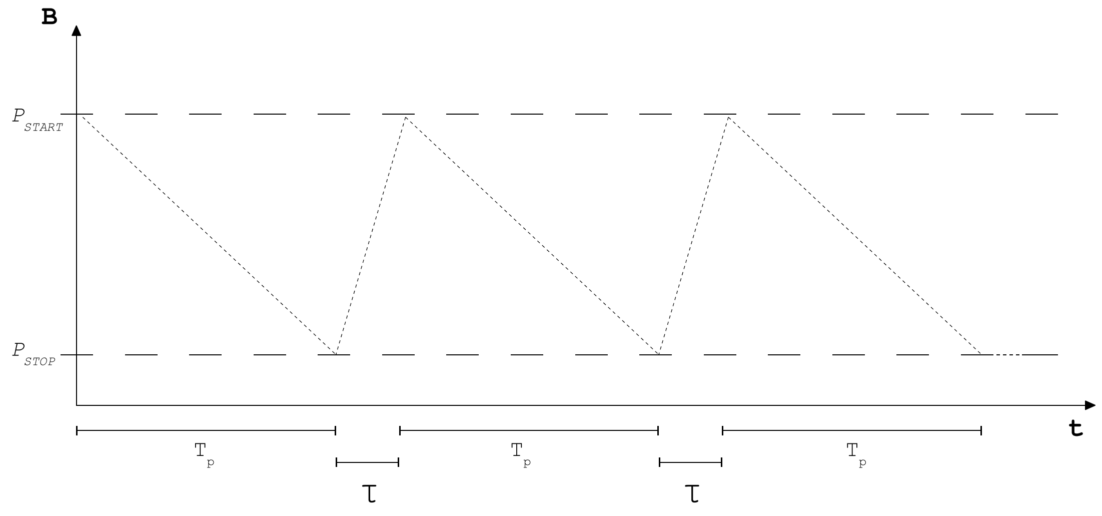


Figure 5.5: Buffer duty cycle when the playback rate is greater than the available throughput.

If V is a discrete random variable representing the pause duration, which attains values $v_1, v_2, \ldots$ with occurrence frequency $f(v_1), f(v_2), \ldots$, the pause intensity can also be obtained by

$$I_p = \sum_{i \in \Phi} [v_i f(v_i)]/T \tag{5.3}$$

where T is the total playback time and Φ is an index set for all the pause durations that have been recorded within T.

Using this metric, it is possible to quantify the effect of pauses that directly hamper the perceived quality in terms of the continuity of playback. As the continuity is closely related to the loss performance of data delivery in a TCP network, it will be shown, through the experiments described below, the relationship between pause intensity and packet loss rates.

5.3 Experiment Setup

The experimental test bed is based on a TCP connection established between two Linux-based machines (a server and a client), using the *iperf* network utility to generate packets and the *Traffic Control* (TC) utility to limit the transmission rate to 1 *Mbit/s*. This closely mimics the traffic pattern of a video being streamed, for example over the *Hyper Text Transfer Protocol* (HTTP), which is typical. A video bit rate of 900*Kbit/s* is selected for the video playback simulator, allowing noticeable observation in changes to the buffer when packet loss is introduced. The network monitoring tool, *tcpdump*, is used to capture and store the timestamp and length of every packet arriving at the *iperf* client. The standard TCP Cubic [82] is used.

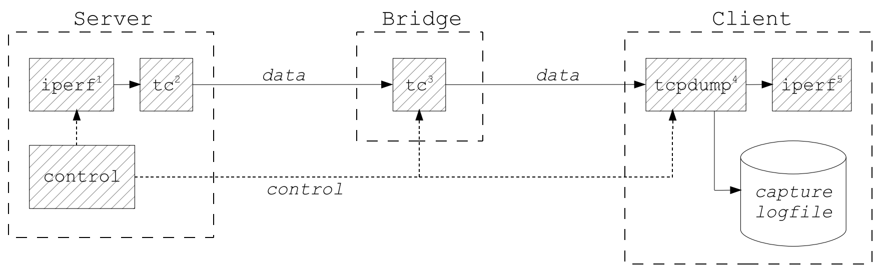


Figure 5.6: The test bed structure: ^{1}traffic generator, ^{2}bandwidth arbitrator, ^{3}packet dropper, ^{4}packet capture and ^{5}traffic endpoint

A third machine is configured using the *tc* utility, bridging the server and client, to drop packets at rates between 0%-10%, incrementing at a 0.5% interval for each capture. A randomly distributed loss algorithm is used to ensure each test is comparable; the average packet loss is calculated based on the entire test duration.

Capture files provided by *tcpdump* are parsed by the simulator to control the buffer of a video playback program growing and shrinking accordingly. With a pre-defined buffer size of 5*Mbit* and a video playback rate close to that of the available bandwidth, it is possible to monitor the utilization of the buffer, the moments when it becomes exhausted and when it recovers. Two thresholds are set in the test bed for playback to start and stop at certain buffer usage levels which indicate the remaining amount of video data cached in the buffer in Mbits.

The now standard TCP Cubic [82] was used with Linux kernel 2.6.23. The machine used for generating packets, limiting bandwidth, parsing the data and simulating had two dual-core Xeon processors running at 3.46GHz and 1GB of RAM. The 1Mbit/s stream was sent to a Pentium III class machine where the data was captured and saved directly to *pcap* format. The bridge was also Pentium III class. Both server, client and bridge use dedicated Intel PRO-based network interfaces running at 100Mbit/s full duplex. Swap space was not utilised in any case, and it was noted that the kernel did not drop any packets during the capture.

5.4 Simulator

Before the simulator is run, the tcpdump packet capture, as outlined in Section 5.3, are decoded to readable ASCII from their binary *pcap* format, and then parsed to form a usable input for controlling the simulator. Listing 5.1 shows the decoded *pcap* output, which is split in to four lines.

Listing 5.1: tcpdump output format

```
23:12:57.229075
IP 10.0.0.1.44338 > 10.0.0.2.3001
:P 52153:53601(1448) ack 1 win 12
<nop,nop,timestamp 481898233 24465804>
```

The first line is the timestamp of the received packet, the second line provides information about the source and destination of the packet. This information is not used in the simulator, however it is used to filter relevant packets when parsing the data. The third line provides sequence numbers and the packet length (bracketed). The final line provides additional information about timing, and is not used. Listing 5.2 shows this data after it has been parsed.

Listing 5.2: Parsed tcpdump data ready for simulator

```
23 12 57 229075 52153 53601 1448
```

The parsed data includes a space-separated timestamp and a packet length field for use with the simulator. The steps within the simulator are shown in Fig. 5.7. The sequencing information is disregarded since the buffer is assumed to deal with any re-ordering issues.

Firstly, in the *Buffer Control* section, buffer B is checked to ensure it is not at full capacity, B_{MAX}, before reading the next packet. If the buffer is full, the packet is ignored and playback is allowed to continue until such a time that B_{MAX} is false. Since the primary functionality of the simulator is to provide data regarding buffer under-runs, discarding packets when the buffer is full is not an issue, since pauses do not and are not expected to occur. The packet length is read and an overhead is subtracted to account for TCP and IP headers before adding

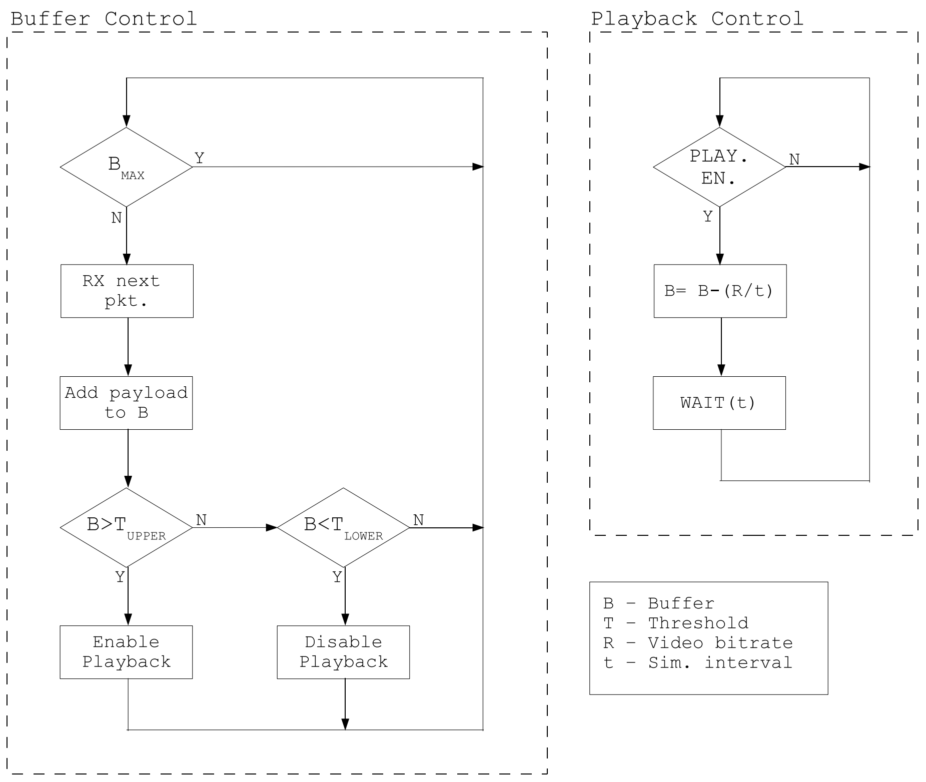


Figure 5.7: The procedure of the simulator

the remaining data to buffer *B*. Checks are made to monitor the size of the buffer, and the playback is enabled or disabled in accordance to the upper and lower thresholds specified.

The *Playback Control* section deducts data from the buffer according to the playback rate specified, and as a fraction of the simulator interval. A delay is placed in this section to ensure the buffer is consumed at the correct rate.

The simulator was written in the C programming language and run in a procedural manner. The main sections can be thought of as running in parallel, although they were coded sequentially within one primary loop. The simulator interval (the rate at which the main loop was executed) was sufficiently higher than both the rate of the incoming data and the rate of video playback, which ensured neither process affected the other. In addition, the *Buffer Control* section is asynchronous of the *Playback Control* delay.

5.5 Results And Analysis

The duration of each simulated video is 1600 seconds, which is around 26 minutes if uninterrupted. In the following figures, the buffer usage is shown for different rates of loss.

Fig. 5.8 demonstrates buffer behaviours during playback when there is no loss (a), and at a loss rate of 1.5% (b), respectively. The two thresholds marked show when playback will be paused (the upper line) and subsequently resumed should the buffer recover (the lower line).

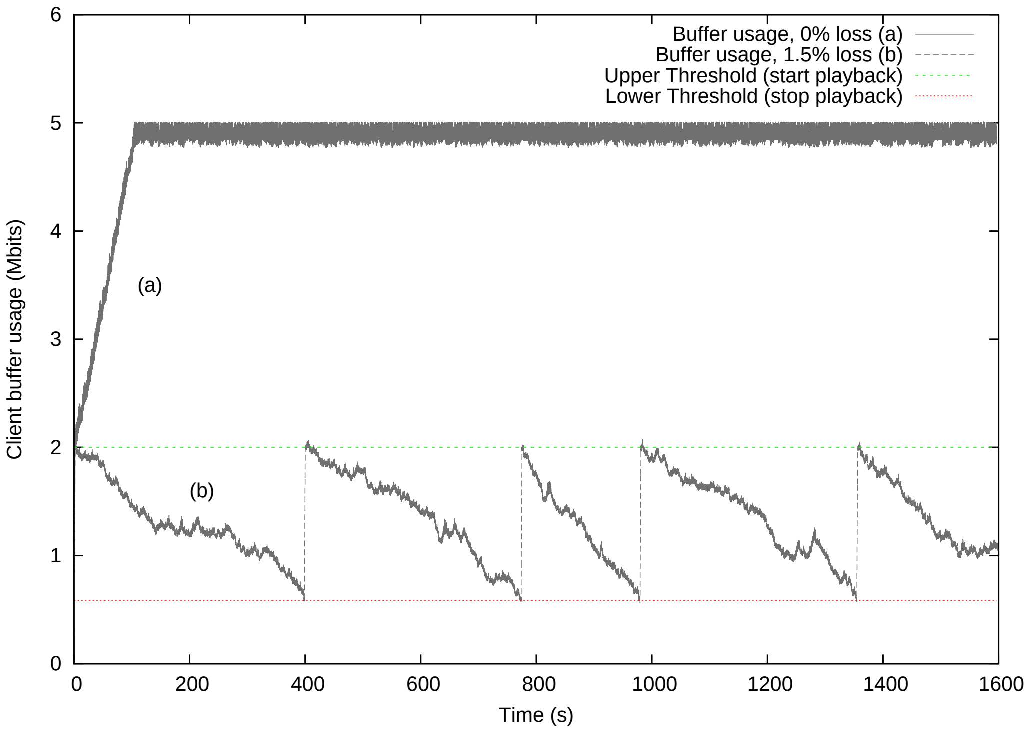


Figure 5.8: Buffer usage at 0% (a) and 1.5% (b) packet loss.

For the case of Fig. 5.8(a), during the stage before playback begins, the buffer fills rapidly since there is no outgoing data at this time. When the upper threshold is met, the buffer fill rate reduces due to playback of video commencing. The buffer continues to fill at this reduced rate until the maximum capacity of the buffer is reached. Since there is no loss, and the bandwidth available is sufficiently higher than the playback rate, the buffer remains full for the duration of the video clip. When the average packet loss rate increases to 1.5% in the case of Fig. 5.8(b), losses cause the input data rate to decrease below the playback rate. As expected, the buffer begins to empty until the lower threshold is met and playback is paused. In this example, the playback pauses on several occasions whilst the buffer recovers.

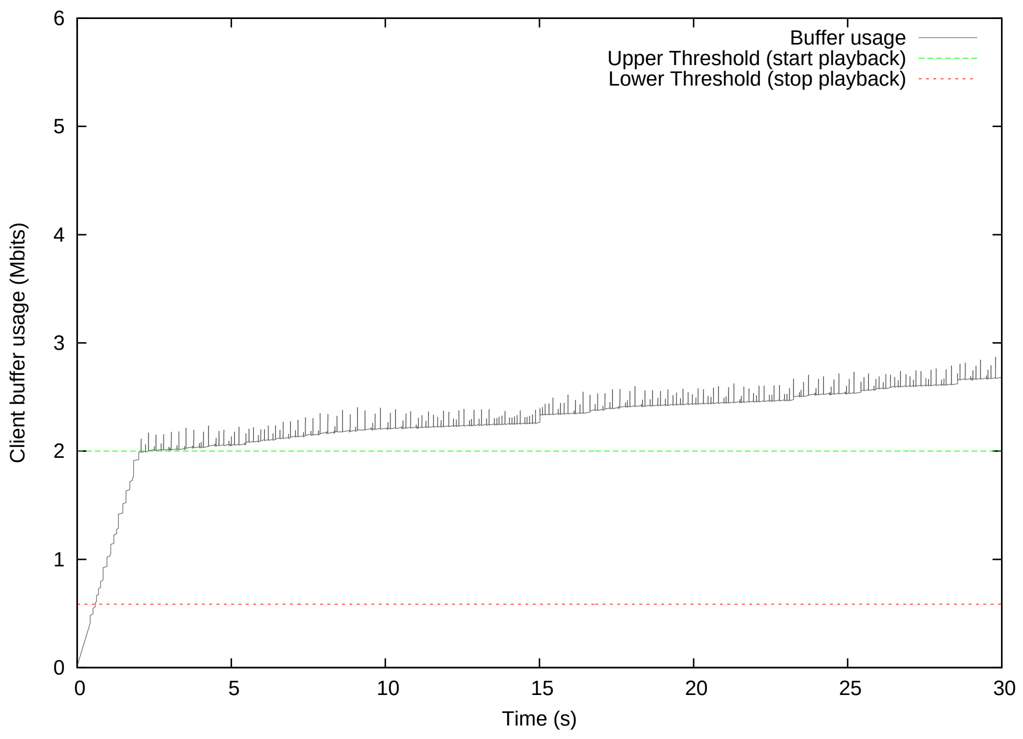
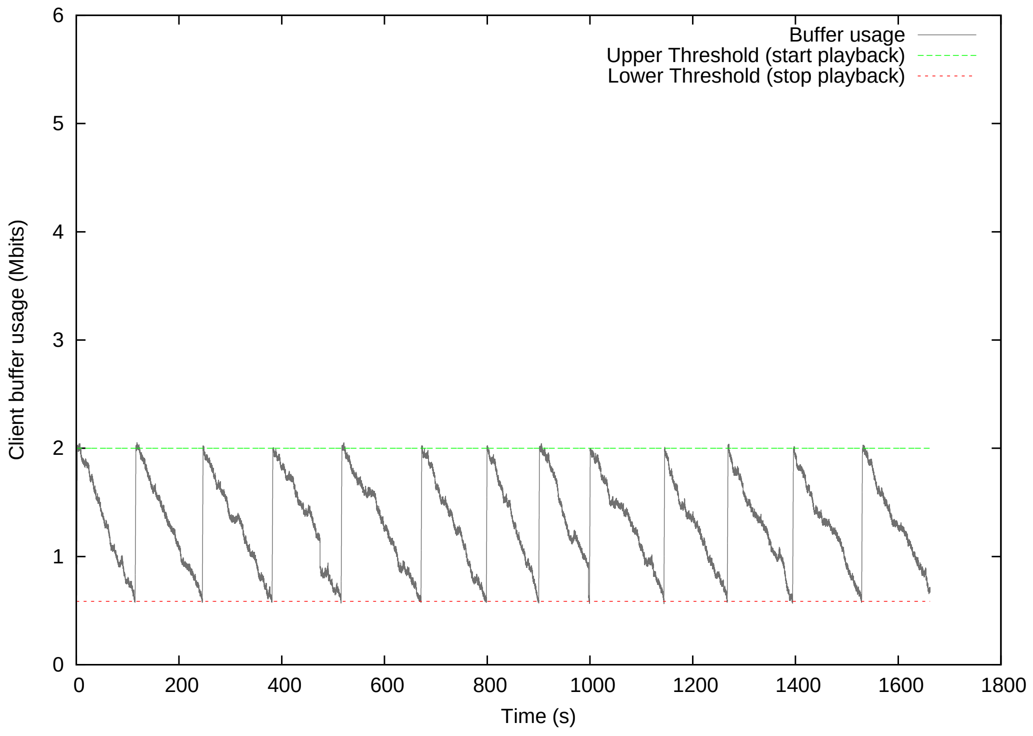


Figure 5.9: The first 30 seconds of buffer usage at at 0% packet loss.

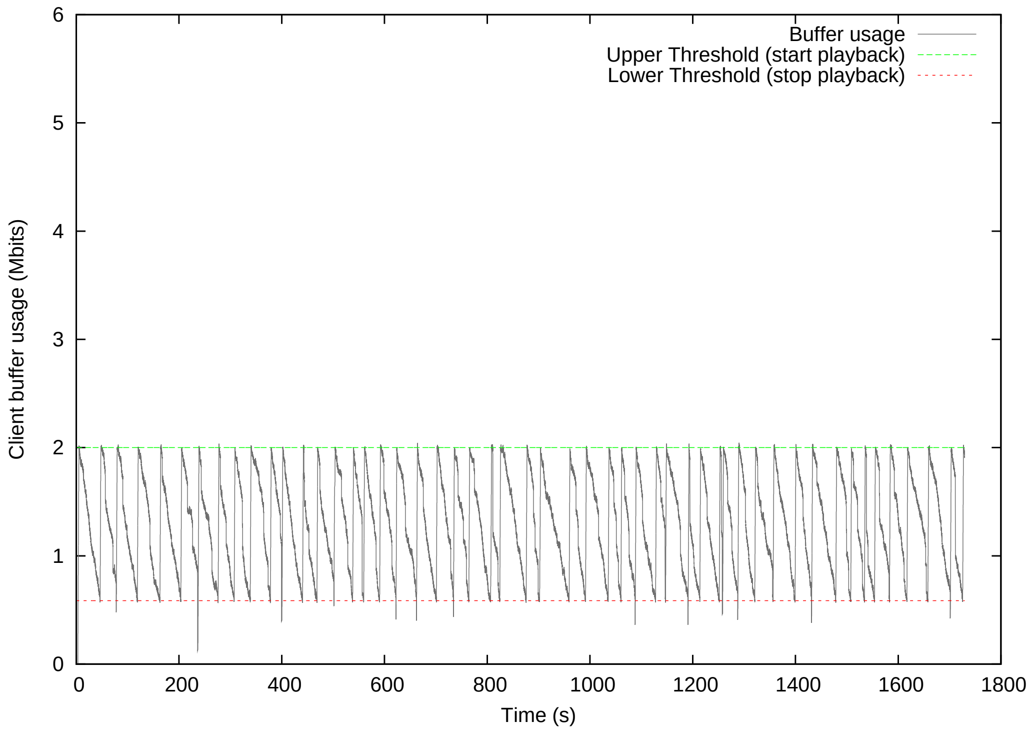
Fig. 5.9 shows the simulated buffer up to $t = 30s$ when packet loss is at zero. During the pre-buffering stage it takes around 2 seconds for the buffer level to reach the upper threshold, at which point playback of the video stream commences. During playback the rate at which the buffer fills decreases, however the buffer to continues to grow due to the ingress of data being marginally higher then the egress. It will grow until the maximum buffer capacity is reached, in this case 5MBit. The occasional spikes on the curve can be explained by varied packet size, primarily due to the TCP Nagle algorithm [15].

Figs. 5.10(a) and 5.10(b) show the fluctuation of the buffer at 2% and 4% packet loss, respectively. For both cases the loss rate is sufficiently high to cause playback to pause for this length of video. In this situation, a reduction in the video bit rate may help to alleviate network congestion and reduce loss, or an increase in buffer size would help initially. The latter approach is only viable for videos which are relatively short in length - for feature length videos such as the one simulated, buffer depletion becomes inevitable.

It is noticed, particularly in Figs. 5.10(a) and 5.10(b), that the buffer usage falls below the lower threshold. These occasions are explained by packet loss occurring at a point when the



(a) 2% packet loss



(b) 4% packet loss

Figure 5.10: Buffer usage at 2% and 4% packet loss.

buffer capacity is very close to the lower threshold. When the next set of bytes are extracted from the buffer, by the *playback control*, it causes the buffer level to drop notably. Playback is then paused after this point. These events become more notable at higher rates of loss.

Collectively, over an extended range of packet loss rates, the number of pauses experienced during playback are determined, along with the duration of the pause in each case.

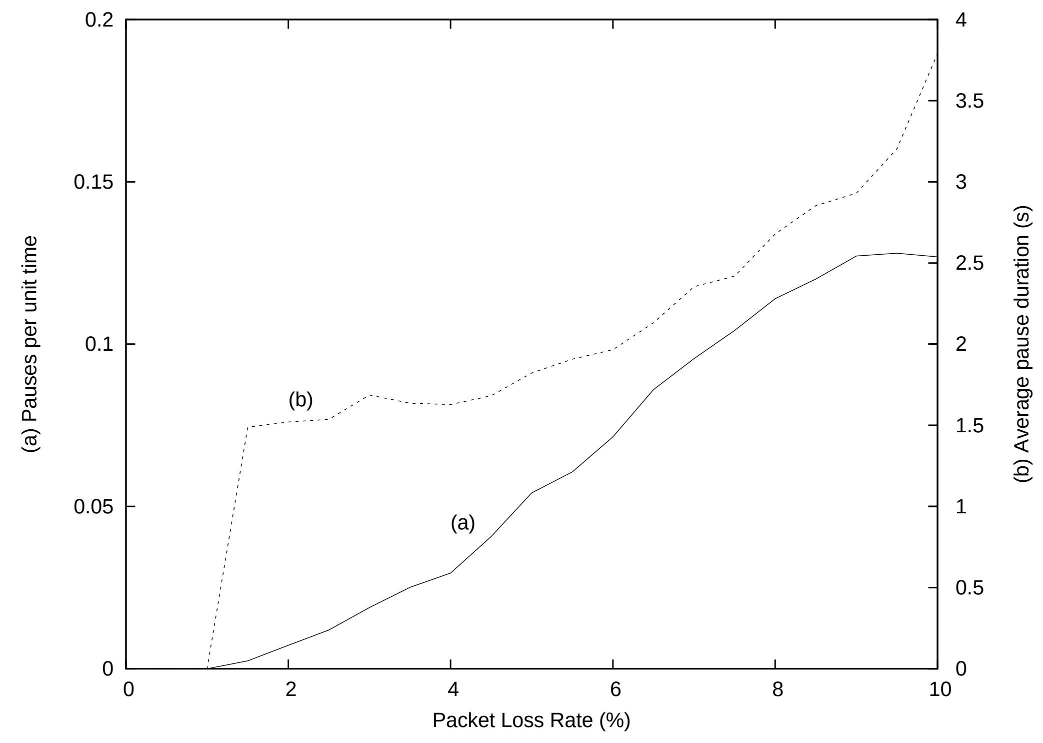
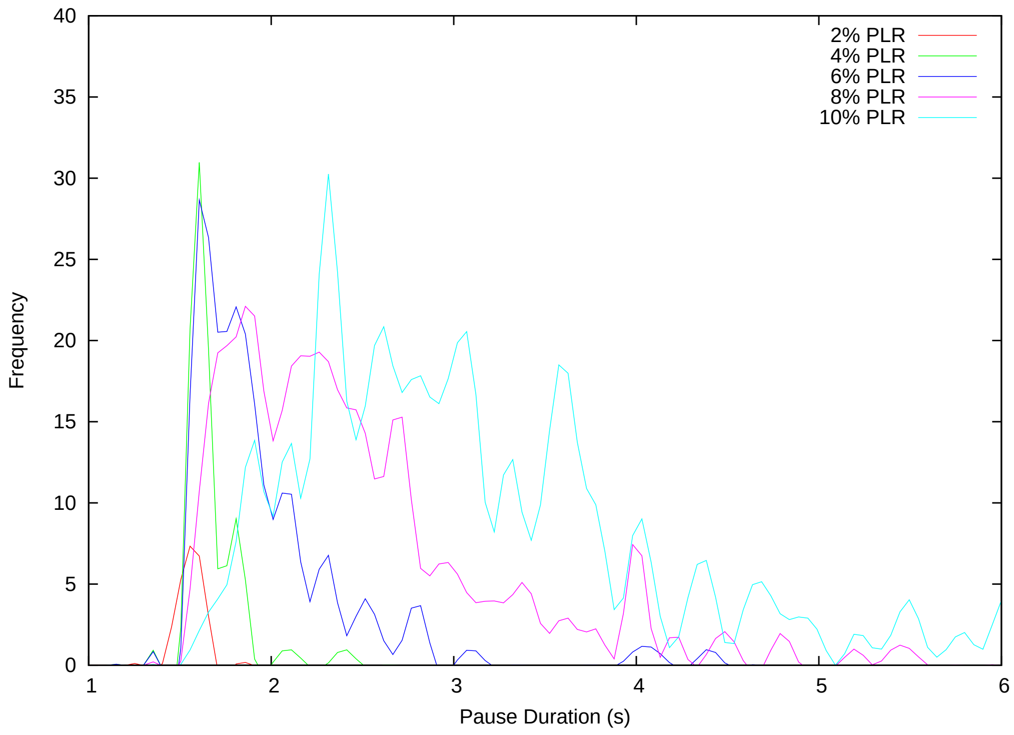


Figure 5.11: Number of pauses (a), and duration of each (b), for each loss rate.

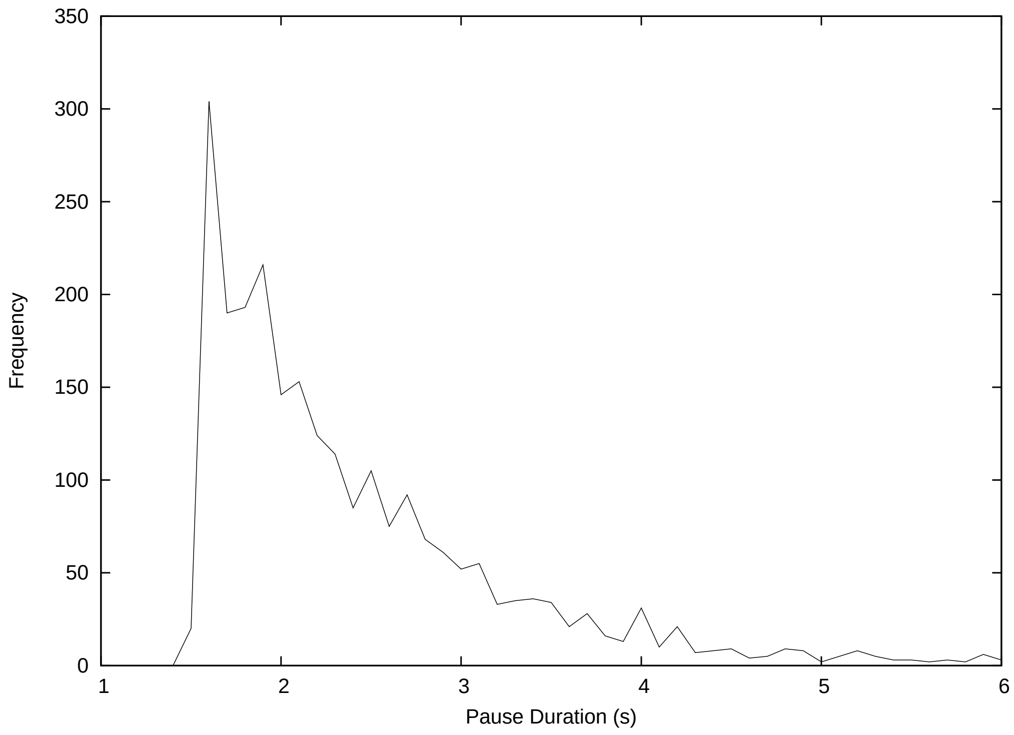
In Fig. 5.11, the number of pauses per unit time (second), u, and the average duration in seconds of each pause, v, are shown for the range of packet loss rates tested by Fig. 5.11(a) and Fig. 5.11(b), respectively. When the packet loss rate is below 1% the buffer is able to maintain playback for the entire duration of video playback with little or no interruption. As the loss rate exceeds 1%, the buffer is seen to deplete on one or more occasions, leading to pauses in playback. For moderate levels of packet loss, e.g. 1%-6%, the duration of each pause is predominantly between 1.5 and 2 seconds, despite the pauses themselves becoming more frequent.

The distributions of pause durations in terms of their occurrence frequency are shown in Fig. 5.12(a) for discrete packet loss rates. It is observed that, when network conditions become far worse, the number of pauses begins to plateau, while the duration of each pause becomes increasingly prolonged. The majority of pause lengths throughout all tests is between 1.5 and 2.5 seconds in Fig. 5.12(a). In addition, Fig. 5.12(b), provides the distribution of pause durations for all loss rates combined.

Based on the result given in Fig. 5.11, pause intensity for individual loss rates can be obtained using either Eq. 5.2 or Eq. 5.3. In particular, at a specific loss rate, the occurrence frequency $f(v_i)$ for each duration value v_i *($i = 1, 2, \ldots$)* can be found from the corresponding distribution graph, e.g. in Fig. 5.12(a). Eq. 5.3 is then applied to calculate the pause intensity value.



(a) Discrete loss rates



(b) Entire experiment

Figure 5.12: The distribution of pause durations, with a bin size of 0.1s

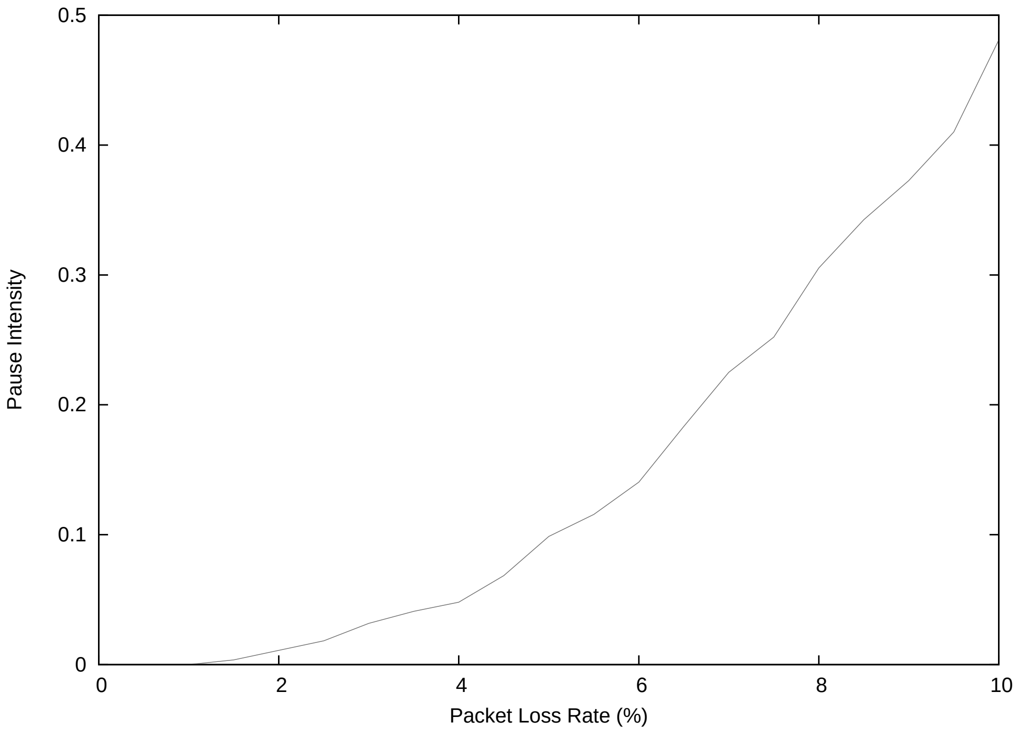


Figure 5.13: The Pause Intensity during playback at different loss rates.

As a result, the relationship between pause intensity and packet loss performance is shown in Fig. 5.13. It can be seen that pause intensity varies monotonically with the packet loss rate across a range of realistic network conditions. This is untrue if just looking at the performance of the number of pauses or pause duration alone as shown in Fig. 5.11.

5.6 Subjective comparison

To further establish the effectiveness of the metric defined, a subjective survey was conducted using a series of videos that were paused at the rates and durations with respect to the packet loss rates in the experimental tests.

The video clips used were full-length film trailers which were chosen due to their rapidly changing content, which requires immediate concentration from the test participant, and kept them engaged throughout the survey. This allowed the use of relatively short video clips and meant the survey lasted around 10-15 minutes, which widened the number of participants. Prior to the viewing tests, a series of questions were asked to ascertain more detail about the survey audience. Socio-demographic information along with their day-to-day viewing of online video content was noted. This data is shown in Figs. 5.14 and 5.15, respectively.

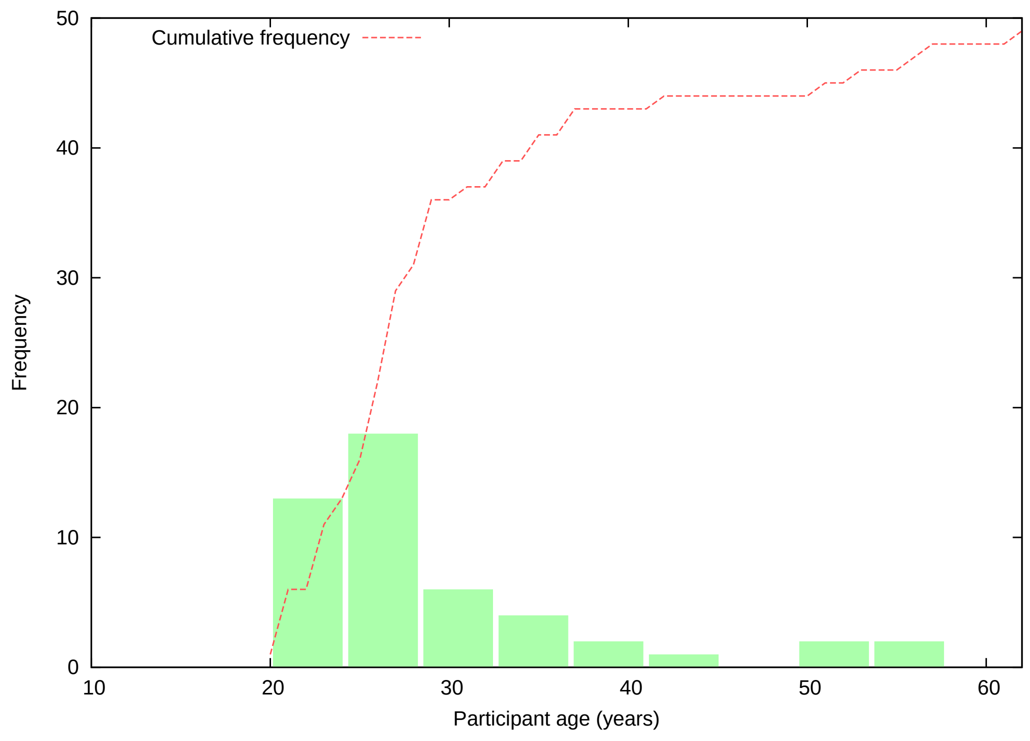


Figure 5.14: The distribution of ages of survey participants

The majority age of the survey participants was between 21 and 30, many of which were students or researchers at Aston University. Friends, colleagues and family members of mixed professional background were also asked to participate. In total, fifty users participated in the survey, which exceeds the minimum sample size of fifteen, as recommended by

the ITU-T P.910 standard for subjective video quality assessments [83].
The two initial questions follow. Firstly, participants were asked:

***(Q1)* "Do you watch any video content online? (tick all that apply)"**

Possible answers included:

- **BBC iPlayer**
- **YouTube**
- **Google Video**
- **Other**

This introductory question was important to confirm that the participants selected had experience watching video online. The specific cases, *BBC iPlayer, Youtube* and *Google Video* all transport video using TCP, so it was assumed that the participants had, at the very least, experienced buffering of video content - either before playback begins or during.
The *Other* option was provided as a default should the first three not apply. In all cases, participants who had selected this option also chose from one of the first three.

The results are shown in Fig. 5.15.

The second part asked of the survey a related, but more specific, question:

(Q2) **”What type of online video content do you watch?”**

Possible answers included:

- **Full, pre-recorded programs** *(eg BBC iPlayer).*
- **Live streamed TV** *(eg BBC live streaming).*
- **Short clips** *(eg YouTube).*

For each part of the answer, participants selected from:

- **Every Day**
- **Most Days**
- **Most Weeks**
- **Seldom**
- **Never**

The type of online video content was divided into three primary categories.

Firstly, *Full, pre-recorded programs* refers to *Video on Demand* (VoD) with a feature length duration, typically 30-60 minutes. The example given in this case was *BBC iPlayer* which provided exactly as described.

Secondly, *Live streamed TV* refers again to VoD, however with the nature of a live schedule. The example given was *BBC live streaming*, for instance online news or viewing of a particular television channel.

Thirdly, *Short clips* such as those found on *Youtube*. These are neither live nor feature-length, but have a typical duration of 1-10 minutes. The results from these questions follow.

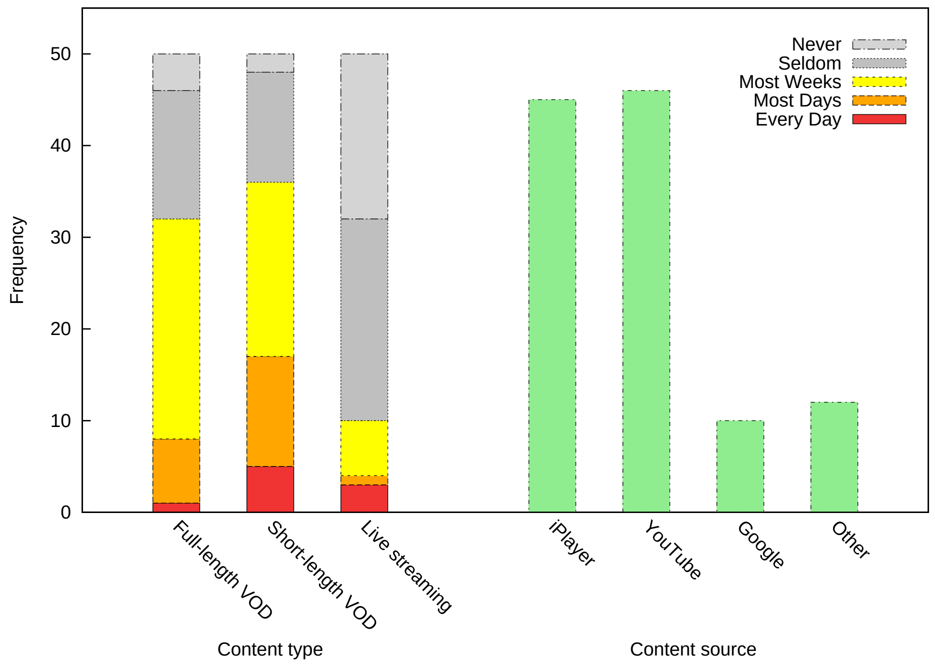


Figure 5.15: Results of the initial subjective survey questionnaire.

For the first question, **”Do you watch any video content online?”**, it was established that 92% of participants have used either *iPlayer* or *YouTube*, and 20% had used Google Video. These results are shown in Fig. 5.15. As mentioned earlier, the 24% who had selected *Other* had also selected one of the first three options.

In question two, participants were asked **”What type of online video content do you watch?”** and, in addition, they were asked how often.

The majority of participants answered *”Most Weeks”* for Full-length VoD and Short-length VoD, at 48%, 38%, respectively. For the case of and Live streaming, the majority 44% seldom watched this kind of content. Only 8% of people asserted they had *Never* watched a Full-length VoD, whilst a lesser 4% is said to have never watched Short-length VoD. However, for Live streaming, over a third of survey participants (36%) claim never to have watched it. How the stream is delivered is not specified; indeed it should be noted that Live streaming is often transported using TCP, since delay of a few seconds is still accepted as being *Live*.

In general, it was found that all participants do, or have in the past, watched at least *some* kind of online video content.

Table 5.1 shows the length of each video used, and the play/pause durations, in seconds.

Video	Length *(s)*	Play dur. *(s)*	Pause dur. *(s)*
1	149	80.0	1.52
2	141	33.9	1.64
3	138	13.8	1.97
4	138	8.77	2.68

Table 5.1: The length and simulated play/pause duration for each video in the survey

The durations are calculated from the data obtained in the experimental procedure, as shown earlier in Fig. 5.11.

To assess the perceptual Quality of Service (QoS) of each video, a *Mean Opinion Score* (MOS) was recorded from each participant throughout the video playback. The MOS system, recommended by the ITU-T, is generally used for assessing the quality of voice in telephony, recently including *Voice over IP* (VoIP). Factors such as distortion, loudness, delay and echo are all covered during tests. For the purpose of this work, the quality factors will be frequency and duration of playback interruptions. In this case the survey is referred to as a *Degradation Category Rating* (DCR) test and participants score using the following 5-point scale: *Imperceptible, Perceptible but not annoying, Slightly annoying, Annoying* and *Very annoying*.

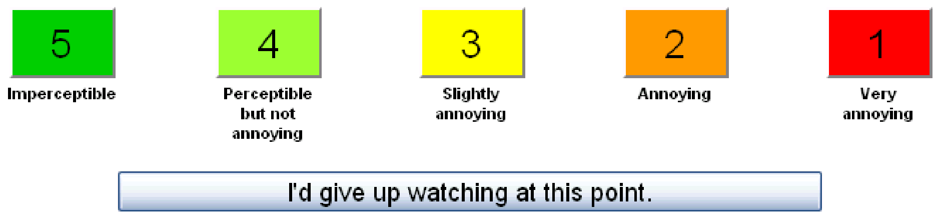

5
Imperceptible
4
Perceptible but not annoying
3
Slightly annoying
2
Annoying
1
Very annoying
I'd give up watching at this point.

Figure 5.16: The method of scoring playback quality.

The arithmetic mean of the results in this case is called the *Degradation MOS*, but will more generally be referred to as the MOS herein. Users were able to score at any time throughout playback, and as many times as they liked. Participants gave scores whilst watching the videos, using five buttons (Fig. 5.16), placed below the video playback window. In addition, there was a button to allow the user to assert a *give up watching* opinion, although the participant was subsequently asked to finish watching the video. After each video had competed, the participants were asked, once again, to provide a score for what they had just seen. The question was structured as ”In reflection, what would be your overall rating of the video you just watched?”. The same scoring panel was provided (Fig. 5.16) as when watching the video, with the exception of the *Give up* option. This question is termed as the *reflective* MOS.

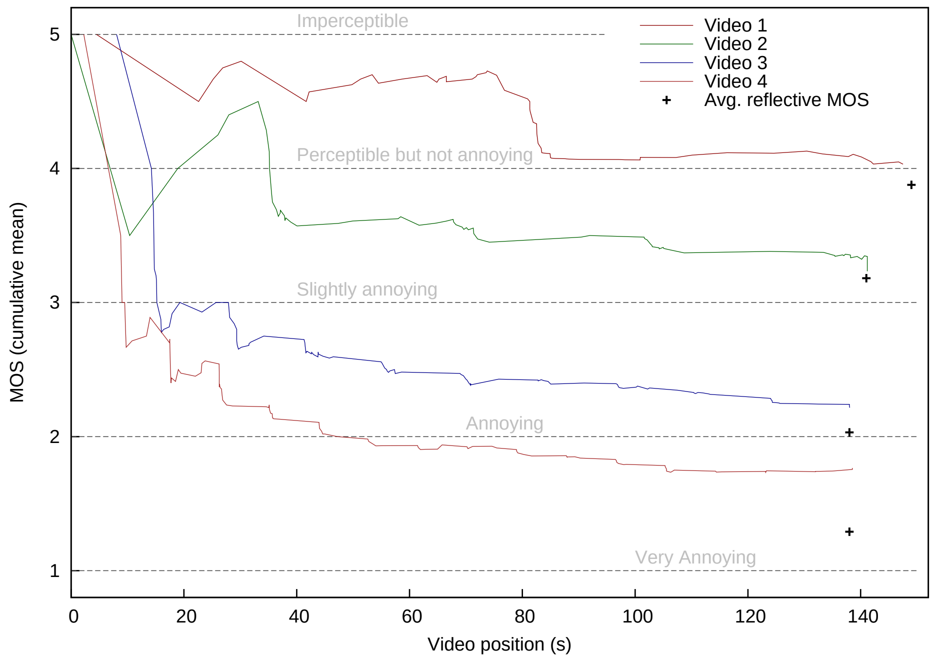


Figure 5.17: The MOS submissions throughout video playback

Fig. 5.17 provides insight to the scoring habits of the participants throughout each of the four video clips. Marked on the graph are the boundaries between each of the 5-point MOS categories. The cumulative mean of the scoring is plotted in each case, and in addition place a cross for the average *reflective* scores.

In each of the four cases, the scoring starts with a high MOS, since participants are able to score at any time during playback. Once the interruption to playback begins, the MOS drops, as expected. For the case of *video 1*, it can be deduced from Table 5.1 that only one interruption occurs throughout playback, which is clearly visible in Fig. 5.17 at around 85 seconds.

In *video 2*, the pauses can be clearly seen occurring around 40, 70 and 100 seconds. A notable attribute in this video is within the first few seconds of playback where the MOS drops rapidly to around 3.5 before returning to a steadier, higher level. This can be explained by the participants flexibility to score at any time, which means the cumulative mean shows only a small percentage of the total scores at very early stages, and should be disregarded. It can be clearly seen in *video 3* and *video 4* that the MOS is significantly reduced when more interruptions to playback are introduced, in accordance to Table 5.1.

In addition to the cumulative mean, the reflective MOS is plotted at the end of each video sequence, marked with a cross. In each case, the reflective MOS is lower than the cumulative mean demonstrating that, in reflection, participants recall the quality of the video playback was worse overall. It is understandable that they do not, at this point, consider the time when playback was smooth, especially at the beginning of each sequence. This is something that the cumulative mean *does* take into account. Both measurements of MOS are considered valid: one provides an averaged scoring over the entire sequence, and the other a final reflective opinion.

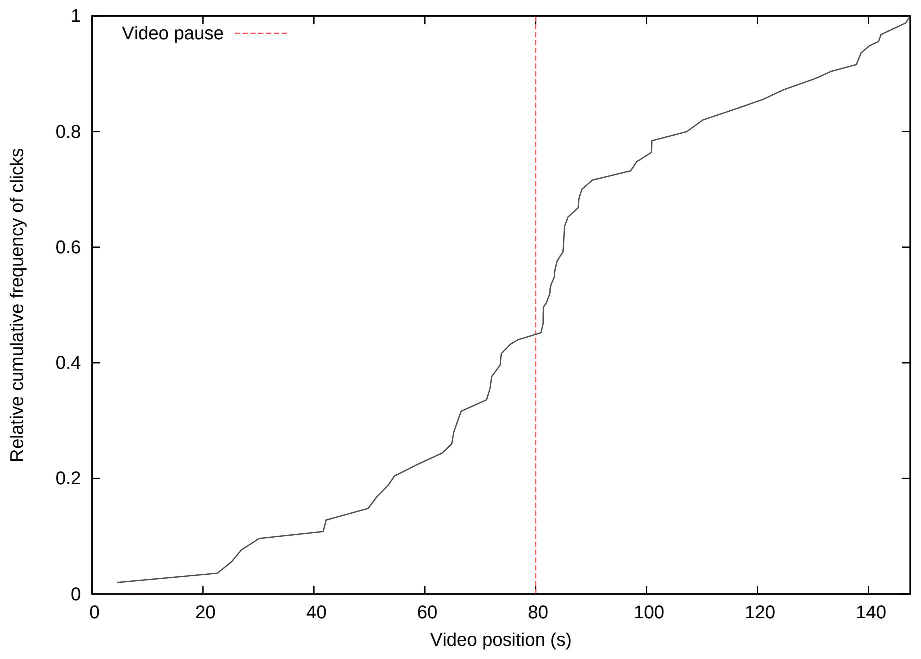
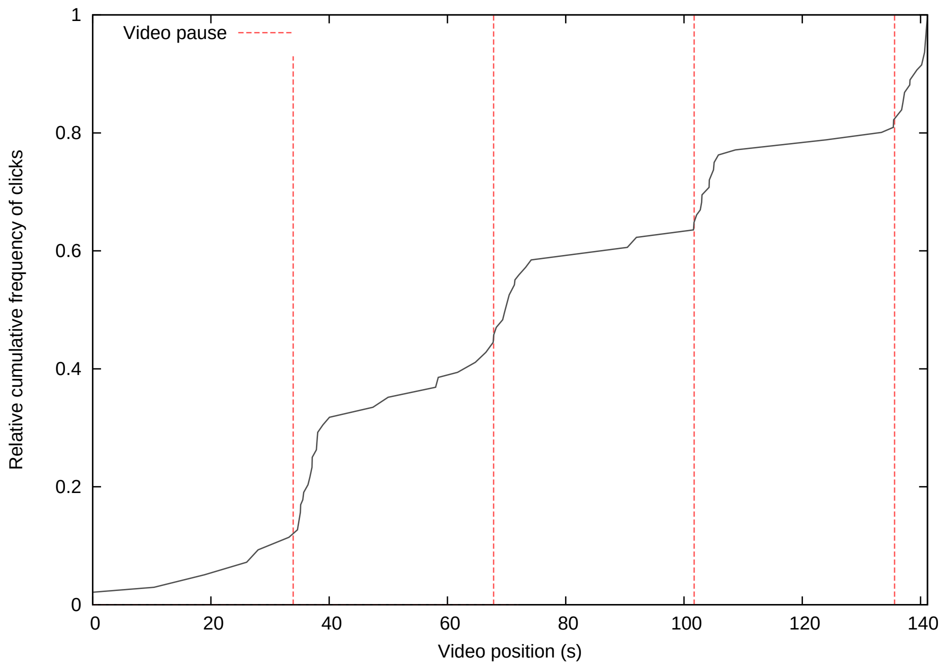
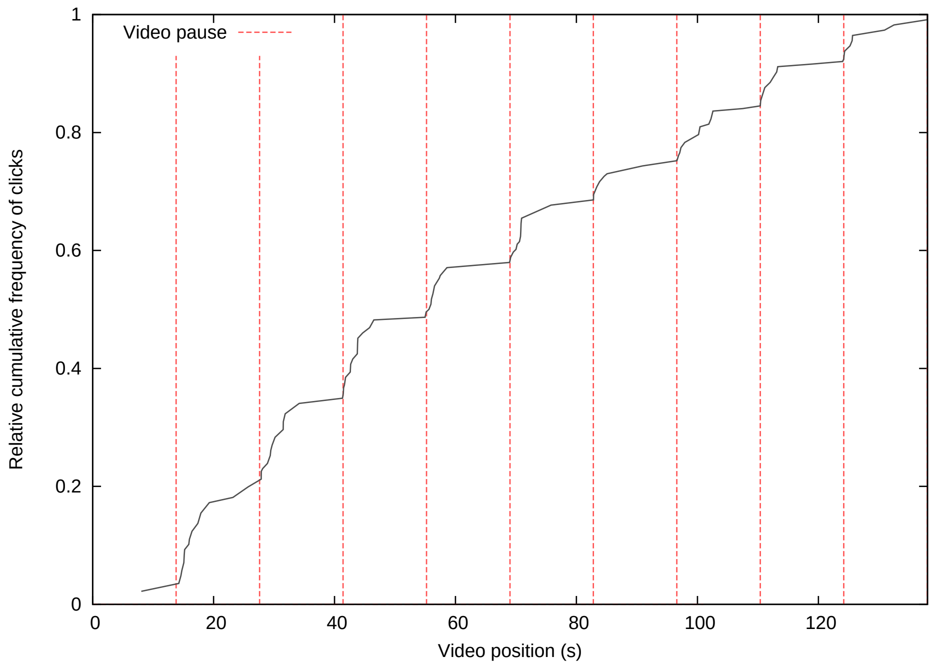


Figure 5.18: The times at which survey participants submitted a score for video 1

Fig. 5.18 provides insight into *when* participants clicked a scoring button. The data is shown as a relative cumulative frequency of clicks against the video playback position. It can be seen how scoring occurs regularly throughout the entire sequence, with notable concentration after a pause in playback. The pause positions are labelled. Graphs for the other three videos are shown in Figs. 5.19(a), 5.19(b) and 5.20.



(a) Video 2



(b) Video 3

Figure 5.19: The times at which survey participants submitted a score.

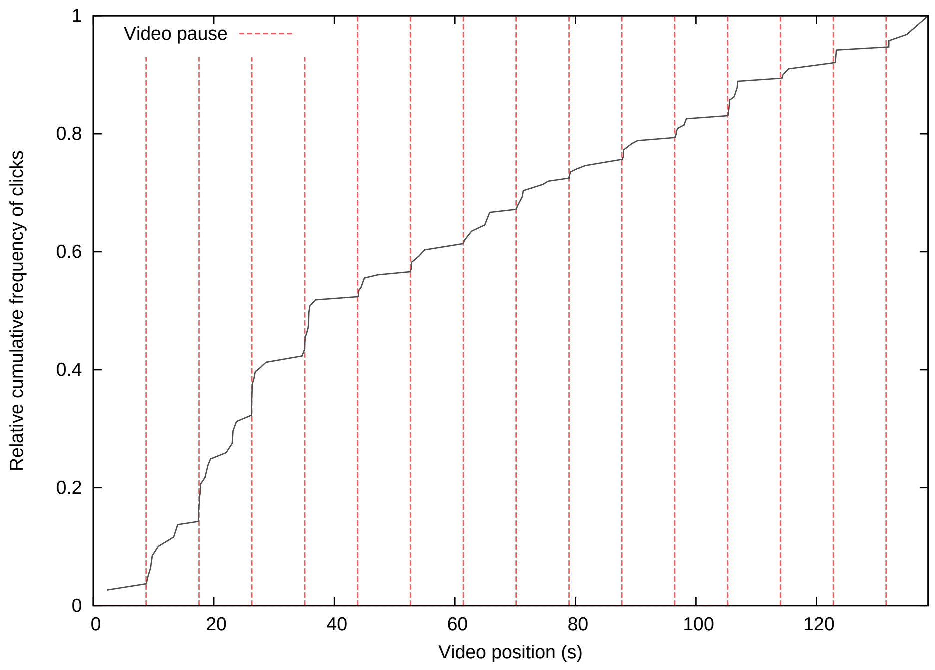


Figure 5.20: The times at which survey participants submitted a score for video 4

After scores were taken for all four video clips, the survey was completed with a further question. Participants were asked:

***(Q4)* ”When experiencing interruptions to playback, like you have just seen, would you tolerate pauses more or less if the video you were trying to watch was feature-length, for example a 30 minute programme?”**

Possible answers included:

- **I would tolerate the same level of interruptions for longer.**
- **I would tolerate the same level of interruptions less.**
- **My tolerance of the interruptions would be the same.**

The results in Fig. 5.21 show that 58% of people asked would tolerate the same level of interruptions less, if the video was feature-length. 18% of participants asserted they would tolerate the interruptions for longer, whilst 24% said they would have the same level of tolerance.

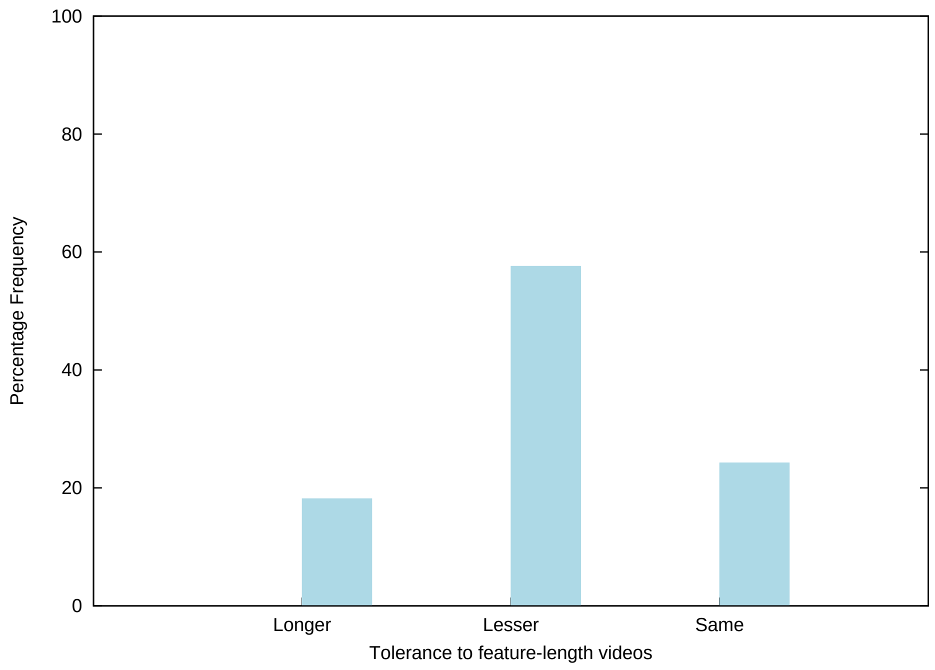


Figure 5.21: How the survey participants would tolerate the interruptions in a feature-length programme.

The results in Table 5.2 provide data on when the participant chose to *give up* watching the video clip because the intensity of pauses were just too great. In the first video, only 4% of participants asserted that they wished to give up watching; the average playback position at this point was 90%. This result can be classed as negligible, since the sample size was so small. Similarly for videos 2 and 3, the sample size is very small; the playback position, on average, was just over half-way through.

For video 4, however, the result is much more significant. 44% of participants asserted they wished to *give up*, on average, at 47% through the video.

Video	Frequency of People	Average Position
1	4%	90%
2	8%	65%
3	12%	64%
4	44%	47%

Table 5.2: The *give up* frequency for each video, and the average position of the video when asserted.

The subjective result against the objective measurement (pause intensity) was based on the same set of videos and is shown in Fig. 5.22.

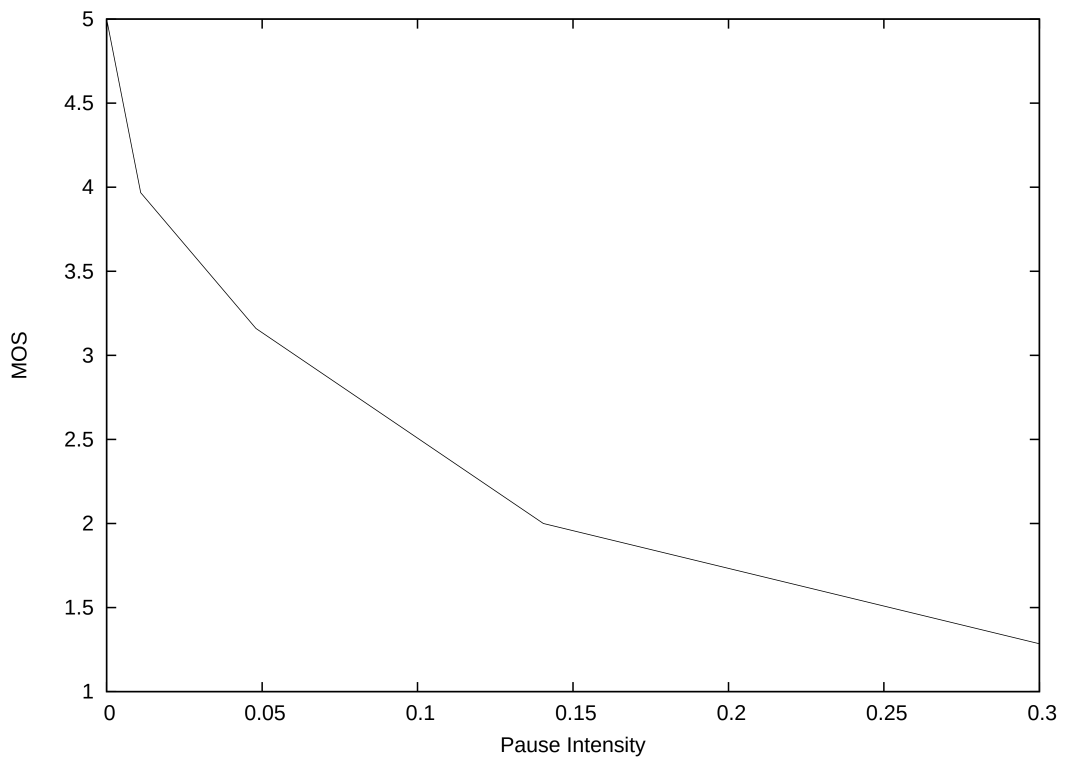


Figure 5.22: The MOS over a range of Pause Intensities.

Clearly, pause intensity is seen to be closely correlated with the subjective measurement, and hence can be used to objectify the quality of playback for video streaming over TCP.

5.7 Conclusion

In this research, a novel method of measuring playback quality is identified when the video is subject to pauses in playback. Since TCP ensures all losses are controlled, video corruption does not occur, and so the existing methods of quality assessment, such as using PSNR, do not apply. The approach described in this work provides an objective measurement using the concept of pause intensity and expresses the quality of video in terms of interruptions, or pauses, in playback. This approach has been adapted from an existing assessment model, traffic intensity, which was used for analogue telephony switching, and is still in use to describe the capacity of a digital system today.

Under lossy network conditions, the results demonstrate typical behaviour of a playback buffer when the data rate of a video is close to that of the maximum bandwidth available, which allows demonstration of the pause intensity quality assessment. The objectively calculated data is quantified by a series of subjective tests, which allows comparison to the renowned MOS scoring system. It is found that the pause intensity metric is closely correlated with the subjective measurement, and hence can be used to objectify the quality of playback for video streaming over TCP.

Chapter 6

Conclusion

6.1 Summary

The work in this Thesis is structured to provide the relevant background required before introducing several research contributions in follow-on chapters. The background provided an in-depth discussion on network infrastructures, video compression, video streaming and channel coding.

It was made clear how losses affected different transport protocols in different ways. For UDP there is no loss recovery available, and so the quality of video playback is degraded when subject to packet loss. For TCP, delivery is guaranteed, and so the degradation in quality is shown in a different manner - pauses in playback. Before this research was conducted, there was no means to objectify the quality loss in terms of interruptions.

In Chapter 3, it was shown how passing data through a coding scheme to protect against loss can improve the throughput of a video stream in comparison to simply allowing TCP to retransmit any lost packets. Recovering any Erasures is far more efficient than requesting the data be sent again, but this must be balanced with the amount of overhead which accompanies such a procedure. Care must be taken when choosing the coding levels to adapt between. The results show how much benefit in throughput one can expect, and whether the additional overhead makes it feasible.

The most significant result is the use of adaptive FEC in a real world test environment achieving an increase in available throughput, compared to using TCP alone. The case study shows how this can be applied practically and, as a result, significantly improve the quality of service experienced by the user, whilst accounting for any limitations.

If the User Datagram Protocol (UDP) were to be used, or selective retransmission adaptation of TCP [63], in an attempt to further improve performance, then the playback of video would become distorted as a result of loss. This is not something users are nowadays accepting of, especially with the impeccable quality of popular playback systems such as the Youtube [12]. It is also worth noting that when using UDP, the congestion control algorithms do not apply, so a large influx of datagrams may cause excessive network congestion.

In addition, this chapter was complimented with a thorough investigation into the implementation of the system using an FPGA. This sets the solid base for which future work in this area can be conducted, and could be used to implement a practical system. The results in this research provide the clear benefits and limitations, for which the future work can be mindful of.

In Chapter 4, a hybrid TCP/UDP approach is presented for transporting H.264 encoded video in error prone networks, and a number of benefits to the proposed system are shown. The system is presented as a compromise between delay-prone TCP and loss-prone UDP, and it is argued that the quality at the receiver often need not be 100% perfect, providing a certain level is assured.

The advantage to using B-frames is demonstrated when encoding an H.264 video stream, and is shown thorough experimental results to yield a benefit in quality of 1.04dB on average, and a 18.6% saving in bandwidth when compared to an encoded video which uses P-frames alone. The quality difference before transmission of the two videos were within 0.07dB. The use of B-frames is made viable using the hybrid approach, because partition A is assured, without the necessity to rely completely upon TCP.

Furthermore, it is shown that the hybrid system does not cause significant levels of delay or jitter, and is shown experimentally that this holds even at moderate levels of packet loss.

The proposed system is compared with other methods of protection, namely FEC. It is shown in one case that for the same bandwidth overhead, FEC is unable to match the performance of the hybrid system in terms of playback quality.

In Chapter 5, a novel method of measuring playback quality is identified when the video is subject to pauses in playback. Since TCP ensures all losses are controlled, video corruption does not occur, and so the existing methods of quality assessment, such as using PSNR, do not apply. The approach described in this work provides an objective measurement using the concept of Pause Intensity and expresses the quality of video in terms of interruptions, or pauses, in playback. This approach has been adapted from an existing assessment model, Traffic Intensity, which was used for analogue telephony switching, and is still in use to describe the capacity of a digital system today.

Under lossy network conditions, the results demonstrate typical behaviour of a playback buffer when the data rate of a video is close to that of the maximum bandwidth available, which allows demonstration of the Pause Intensity quality assessment. The objectively calculated data is quantified by a series of subjective tests, which allows comparison to the renowned MOS scoring system.

It is found that the pause intensity metric is closely correlated with the subjective measurement, and hence can be used to objectify the quality of playback for video streaming over TCP.

6.2 Personal Contribution

The key points of the summary and personal contributions are specifically outlined below.

Combining FEC with TCP

The use of FEC in combination with TCP has shown to significantly decrease the number of re-transmissions required to ensure delivery of data. This ensures throughput remains at a more usable level because the TCP window size is not so drastically affected. The collection of data in this work was achieved by strict experimental procedure, which is detailed in full. The advantages to this result is shown through a case study to clearly demonstrate the benefits in a real world application. This work has been published in IEEE International Symposium on Broadband Multimedia Systems and Broadcasting.

Hybrid TCP/UDP Transport

A new method of video transport is presented to highlight the advantages of using a hybrid TCP/UDP approach for unequal loss protection. Being able to guarantee partition A slices in a data partitioned H.264 stream allows use of B-frames which not only improves quality, but reduces the data rate. For the streaming environment, this is a significant achievement. This work has been accepted for publication in IEEE International Conference on Multimedia and Expo 2011.

Pause Intensity

A novel metric is defined which allows the assessment of video quality based on its playback continuity. This objective measurement is further validated by correlation with subjective measurement. This significant new metric provides the ability to objectively assess the quality of video playback, if the content is being transported using TCP. This work has been published in the IEEE Communications Letters journal publication, Volume 15, Issue 1.

MOS testing

A substantially scaled survey was conducted to validate the novel pause intensity metric, which provides vast insight into current viewing habits and how accepting people are to pauses in playback. It was shown how many participants chose to simply give up watching the test clips when the pause intensity reached a certain level. The data collected is invaluable research, especially in today's era, because many of the websites and services which offer streaming video is subject to the pause intensity problem.

6.3 Future Work

The most significant area of future work to be conducted is related to hardware implementation of the TCP/FEC system. The area was investigated in depth, and a proposed network solution outlined using a hardware-based FPGA design, based on the NetFPGA platform. A *Finite State Machine* (FSM) was produced to describe the functionality of the design. The FSM can be used to create the necessary HDL code to generate a top-level sequential logic circuit, which will drive a complete system. Once developed in hardware, the efficiency of the encode/decode process is expected to outperform that of a software implementation, which is important for scalability. In addition, all existing hardware can be used without changes to the software or system configuration, since the TCP/FEC layer is designed to be transparent.

In other areas, there is potential for using the newly defined pause intensity metric as an objective measurement tool in further research, such as the hybrid approach, since pauses are likely to occur at high rates of packet loss. Since the trend of pause intensity and MOS is shown to be closely correlated, use of the metric can be taken as a gauge of subjective opinion, without the requirement to conduct further, time-consuming MOS surveys.

References

[1] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modeling of the temporal dependence in packet loss. *IEEE International Conference on Computer Communications*, pages 345–352, 1999.

[2] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[3] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[4] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[5] A. Tanenbaum. *Computer Networks*. Prentice Hall, 2002.

[6] A.C. Bovik, editor. *Handbook of Image and Video Processing*. Academic Press, 2000.

[7] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer. RTP Payload Format for H.264 Video. RFC 3984 (Proposed Standard), February 2005.

[8] X.-J. Zhang, X.-H. Peng, D. Wu, T. Porter, and R. Haywood. *IEEE Consumer Communications and Networking Conference*.

[9] S.W. McLaughlin and D. Warland. Error controling and ethernet. *Calimetrics Inc. Tech. Report*, page 13, 2001.

[10] T. Porter and X.-H. Peng. Effective video content distribution by combining TCP with adaptive FEC coding. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2010.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFC 5506.

[12] Youtube website [available online]. `http://www.youtube.com` .

[13] BBC iPlayer website [available online]. `http://www.bbc.co.uk/iplayer` .

[14] J.D. Day and H. Zimmermann. *Conformance testing methodologies and architectures for OSI protocols*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[15] J. Nagle. Congestion control in IP/TCP internetworks. RFC 896, January 1984.

[16] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), September 1981. Updated by RFCs 950, 4884.

[17] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[18] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[19] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348 (Proposed Standard), September 2008.

[20] G.J. Sullivan A. Luthra and T. Wiegand. Special issue on H.264/AVC. *IEEE Transactions on Circuits and Systems on Video Technology*, 2003.

[21] Nokia position paper. "H.261, which (in its first version) was ratified in november 1988.". `http://www.w3.org/2007/08/video/positions/Nokia.pdf` , 2007.

[22] H. Benoit. *Digital Television: MPEG-1, MPEG-2 and Principles of the DVB System.* Focal Press, 2002.

[23] S. Wenger. H.264/avc over ip. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):645 – 656, jul. 2003.

[24] K. Jack. *Video Demystified: A Handbook for the Digital Engineer.* HighText Publications, 5 edition, 2007.

[25] J. Klaue, B. Rathke, and A. Wolisz. Evalvid - a framework for video transmission and quality evaluation. In *In Proc. of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 255–272, 2003.

[26] S.B. Wicker. *Error control systems for digital communication and storage.* Prentice Hall, 1995.

[27] O. Hillestad, O. Jetlund, and A. Perkis. Rtp-based broadcast streaming of high definition h.264/avc video: An error robustness evaluation. *Journal of Zhejiang University - Science A*, 7:19–26, 2006. 10.1631/jzus.2006.AS0019.

[28] European Telecommunications Standards Institute. Digital broadcasting system for television sound and data services. *ETS 200 421*, Dec 1994.

[29] D. Barman, I. Matta, E. Altman, and R. Azouzi. Tcp optimization through fec, arq and transmission power tradeoffs. 2004.

[30] A. Sathiaseelan and T. Radzik. *Improving the Performance of TCP in the Case of Packet Reordering*, volume 3079. Springer Berlin / Heidelberg, 2004.

[31] IEEE-SA. Ieee 802.3: Ethernet specifications. *The IEEE 802.3 Working Group.*

[32] R.C. Crane and E.A Taft. Practical considerations in ethernet local network design. *Proc. of the 13th Hawaii Internat. Conf. on Systs. Sci.*, pages 166–174, 1980.

[33] IEEE-SA. Ieee 802.11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *The IEEE 802.11 Working Group*.

[34] T.K. Moon. *Error Correction Coding*. John Wiley & Sons, 2005.

[35] P. Sweeney. *Error control coding: from theory to practice*. Wiley-Blackwell, 2002.

[36] S.Lin and D.J. Costello Jr. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[37] J.L. Massey and D.J. Costello Jr. Nonsystematic convolutional codes for sequential decoding in space applications. *IEEE Transactions on Communication Technology*, 19(5), 1971.

[38] R.E. Blahut. *Algebraic codes for data transmission*. Cambridge University Press, 2003.

[39] R. Johannesson and K.S. Zigangirov. *Fundamentals of Convolutional Coding*. Wiley-IEEE Press, 1999.

[40] T. Nguyen and A. Zakhor. Distributed video streaming with forward error correction, April 2002.

[41] Y. Xu and T. Zhang. Variable shortened-and-punctured reed-solomon codes for packet loss protection. *Broadcasting, IEEE Transactions on*, 48(3):237 – 245, September 2002.

[42] T. Zhang and Y. Xu. Unequal packet loss protection for layered video transmission. *IEEE Trans. Broadcast*, 45:243–252, 1999.

[43] H. Wu, M. Claypool, and R. Kinicki. Adjusting forward error correction with quality scaling for streaming mpeg. pages 111–116, 2005.

[44] S. Mamidi, M.J. Schulte, D. Iancu, A. Iancu, and J. Glossner. Instruction set extensions for reed-solomon encoding and decoding. In *Application-Specific Systems, Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on*, pages 364 – 369, 2005.

[45] J.A. Cooley, J.L. Mineweaser, L.D. Servi, and E.T. Tsung. Software-based erasure codes for scalable distributed storage. In *Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, pages 157 – 164, 2003.

[46] D.J. Costello Jr., J. Hagenauer, H. Imai, and S.B. Wicker. Applications of error-control coding. *Information Theory, IEEE Transactions on*, 44(6):2531–2560, Oct 1998.

[47] Y. Wu and B. Caron. Digital television terrestrial broadcasting. *Communications Magazine, IEEE*, 32(5):46–52, May 1994.

[48] V.K. Bhargava S.B. Wickerand. *Reed-Solomon Codes and Their Applications*. Wiley-IEEE Press, 1999.

[49] K.J. Kerpez. Forward error correction for asymmetric digital subscriber lines (adsl). *Global Telecommunications Conference, 1991. GLOBECOM '91. 'Countdown to the New Millennium. Featuring a Mini-Theme on: Personal Communications Services*, pages 1974–1978 vol.3, Dec 1991.

[50] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. 1998.

[51] J. Lacan, L. Lancérica, and L. Dairaine. When fec speed up data access in p2p networks. In *IDMS/PROMS 2002: Proceedings of the Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems*, pages 26–36, London, UK, 2002. Springer-Verlag.

[52] X.-J. Zhang, X.-H. Peng, T. Porter, and R. Haywood. Robust video transmission over lossy network by exploiting h.264/avc data partitioning. In *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, pages 307–314, Sept. 2008.

[53] V. Jacobson and M.J. Karels. Congestion avoidance and control, 1988.

[54] C. Barakat and E. Altman. Bandwidth tradeoff between tcp and link-level fec. *Computer Networks*, 39:133–150, 2002.

[55] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[56] X.-H. Peng. Erasure-control coding for distributed networks. *Communications, IEE Proceedings-*, 152(6):1075–1080, Dec. 2005.

[57] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999. Updated by RFC 3390.

[58] I. Rhee and L. Xu. Cubic: A new tcp-friendly high-speed tcp variant.

[59] M.J. Thul and N. Wehn. Fpga implementation of parallel turbo-decoders. pages 198 – 203, 2004.

[60] Xilinx, inc. website [available online]. `http://www.xilinx.com` .

[61] G. Gibb, J.W. Lockwood, J. Naous, P. Hartke, and N. McKeown. Netfpga - an open platform for teaching how to build gigabit-rate network switches and routers. *Education, IEEE Transactions on*, 51(3):364 –369, 2008.

[62] J.W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo. Netfpga - an open platform for gigabit-rate network switching and routing. pages 160 –161, 2007.

[63] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), October 1996.

[64] T. Porter and X.-H. Peng. An objective approach to measuring video playback quality in lossy networks using tcp. In *IEEE Communications Letters*, 2010.

[65] Y.J. Liang, J.G. Apostolopoulos, and B. Girod. Analysis of packet loss for compressed video: does burst-length matter? *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 5:V – 684–7 vol.5, apr. 2003.

[66] X.-J. Zhang, X.-H. Peng, T. Porter, and R. Haywood. Robust video transmission over lossy network by exploiting h.264/avc data partitioning. pages 307 –314, 2008.

[67] N. Thomos, S. Argyropoulos, N.V. Boulgouris, and M.G. Strintzis. Robust transmission of h.264/avc video using adaptive slice grouping and unequal error protection. *Multimedia and Expo, 2006 IEEE International Conference on*, pages 593 –596, jul. 2006.

[68] E.N. Gilbert. Capacity of a burst-noise channel, bell syst. *Tech. J*, (39):1253–1265, 1960.

[69] H. Arachchi, W. Fernando, and S. Panchadcharman. Unequal error protection technique for roi based h.264 video coding. *IEEE Canadian Conference on Electrical and Computer Engineering*, 2006.

[70] R. Haywood and X.-H. Peng. On packet loss performance under varying network conditions with path diversity. pages 1–7, 2008.

[71] ITU-T Recommendation for H.264. *Audiovisual And Multimedia Systems*. 2005.

[72] J. Emigh. New flash player rises in the web-video market. *Computer*, 39(2):14, 2006.

[73] VLC media player [available online]. `http://www.videolan.org/` .

[74] A. Luthra and P.N. Topiwala. Overview of the H.264/AVC video coding standard. *Applications of Digital Image Processing XXVI*, 5203(1):417–431, 2003.

[75] T. Porter and X.-H. Peng. Effective video content distribution by combining TCP with adaptive FEC coding. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2010.

[76] Q. Huynh-Thu and M. Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics Letters*, pages 800–801, 2008.

[77] J. Afzal, T. Stockhammer, T. Gasiba, and W. Xu. Video streaming over mbms: A system design approach. *Journal Of Multimedia*, 1(5):25–25, August 2006.

[78] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia streaming via TCP: An analytic performance study. *ACM Transactions on Multimedia Computing, Communications and Applications*, 4(2):16:1–16:22, May 2008.

[79] R.L. Freeman. *Telecommunication System Engineering*. Wiley-Interscience, 4 edition, 2004.

[80] R.R. Mina. *Introduction to Teletraffic Engineering*. Chicago, Telephony Pub Co., 1974.

[81] J.W. Roberts. Traffic theory and the internet. *IEEE Communications Magazine*, January:94–99, 2001.

[82] S. Ha, I. Rhee, and L. Xu. Cubic: a new TCP-friendly high-speed TCP variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, 2008.

[83] ITU-T recommendation P.910. Subjective video quality assessment methods for multimedia applications., 1999.

Appendix A
Publications

- **"Robust video transmission over lossy network by exploiting H.264/AVC data partitioning"**

  Xing-Jun Zhang, Xiao-Hong Peng, Timothy Porter and Richard Haywood.

  IEEE International Conference on Broadband Communications, Networks and Systems, 2008.

- **"A Hierarchical Unequal Packet Loss Protection Scheme for Robust H.264/AVC Transmission"**

  Xing-Jun Zhang, Xiao-Hong Peng, Dajun Wu, Timothy Porter and Richard Haywood.

  IEEE Consumer Communications and Networking Conference, 2009.

- **"Effective video content distribution by combining TCP with adaptive FEC coding"**

  Timothy Porter and Xiao-Hong Peng

  IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2010.

- **"An Objective Approach to Measuring Video Playback Quality in Lossy Networks using TCP"**

  Timothy Porter and Xiao-Hong Peng

  IEEE Communications Letters, 2011, Volume 15, Issue 1.

- **"Hybrid TCP/UDP Video Transport For H.264/AVC Content Delivery in Burst Loss Networks"**

  Timothy Porter and Xiao-Hong Peng

  IEEE International Conference on Multimedia and Expo, 2011