A source driven adaptation technique for streaming of MPEG video over IP networks

Mohammad Reza Salehi

A Thesis

In The Department

Of

Electrical and Computer Engineering

Presented in partial fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

December 2002

0-612-77690-5

ABSTRACT

A source driven adaptation technique for streaming of MPEG video over IP networks

Mohammad Reza Salehi

Streamed MPEG video over IP networks, like any other information sent over "best effort" networks, is susceptible to loss and unpredictable delays. However, in MPEG video, loss and unacceptable delays are propagated to other frames due to Intra/Inter frame encoding nature of compression. Different adaptation techniques are used to enhance Quality of Service and provide "controlled" degradation on the streamed video.

In this research we consider a source driven adaptation technique through a detailed simulation. Then we propose two schemes for encapsulation of MPEG video inside RTP/UDP/IP packets consistent with the standard and examine how this affects the network level QoS of the video before and after adaptation. A comparison between the two proposed encapsulation methods will also be done from bandwidth efficiency point of view. Effect of different network parameters on QoS for the both suggested encapsulations will be investigated too.

ACKNOWLEDGEMENTS

During the whole process of this study I received help and support from my supervisor Dr. Ahmed El-Hakeem, whom not only proposed the research topic, but also patiently provided continues support and directing this work. I am sincerely grateful to him.

I also would like to thank my wife Laleh, for her support throughout this work. In fact, she was the one who encouraged me to write the simulation in C++ and made valuable comments during the simulation program. I am also thankful to my daughter Yassaman for her patience during the period of my studies. I am grateful to both of you.

Table of Contents

1 Introduction **1**

1.1 Transmission of MPEG video over IP networks 1
1.2 Problem Statement 2
1.3 Organization of the Thesis 3

2 A quick look at IP networks and MPEG video **4**

2.1 Part one 4
2.1.1 Real time applications and IP networks 4
2.1.1.1 Application-level and Network-level interconnection 4
2.1.1.2 Internet Protocol (IP) 5
2.1.1.2.1 Fragmentation and Re-assembly 7
2.1.1.2.2 Delay of Datagrams 8
2.1.1.3 Routers 9
2.1.1.3.1 Queuing in the routers 10
2.1.1.3.2 Random Early Discard (RED) 12
2.1.1.4 User Datagram Protocol (UDP) 14
2.1.2 Real time applications over IP 14
2.1.2.1 Real-Time Transport Protocol (RTP) 15
2.1.2.2 Quality of Service (QoS) 17
2.1.2.3 Integrated Services (RSVP) 18
2.1.2.4 Differentiated Services 21
2.2 Part Two 24
2.2.1 A quick look at MPEG video 24
2.2.1.1 Video MPEG 24
2.2.1.2 MPEG Frames 24
2.2.1.3 Intraframe encoding 25
2.2.1.4 Quantization 26
2.2.1.5 Predicted coding 28
2.2.1.6 Bi-directional Prediction 29
2.2.2 MPEG video structure 29
2.2.3 Dependency of the frames in MPEG GoP 30
2.2.4 MPEG-1 and MPEG-2 31
2.2.5 Variable Bit Rate (VBR) and constant Bit Rate (CBR) encoding 32
2.2.6 MPEG-2 Systems Layer 33

3 Issues on transmitting of MPEG video over IP networks **36**

3.1 Adaptive Streaming and Filtering 36

3.1.1 Server generated degradation 37
3.1.2 Network generated degradation 37
3.1.3 Receiver generated degradation 38
3.1.4 Adaptation techniques 38
3.1.4.1 Receiver Driven Adaptation 38
3.1.4.2 Source Driven Adaptation 39
3.2 Error Propagation in MPEG video stream 40
3.2.1 Syntactic Error Propagation 40
3.2.2 Semantic error propagation 41
3.2.3 Error Concealment Techniques 42
3.3 Packetization of MPEG streams in RTP payloads 43
3.3.1 RTP Encapsulation of MPEG-2 Transport Stream 44
3.3.2 RTP Encapsulation of MPEG-2 Elementary Stream 45
3.4 Network Performance Measures 46

4 A source driven adaptation technique 48

4.1 Investigating packetization schemes within the adaptation techniques 48
4.1.1 Figure of merit 48
4.2 The adaptation technique 49
4.2.1 Frame dropping 49
4.2.2 Changing quantizer step size 53
4.2.3 Packetization 54
4.2.4 Effects of networks parameters 54

5 Simulation and results 55

5.1 Program 55
5.2 Simulation Assumptions 56
5.2.1 Traffic sources 57
5.2.1.1 Internet Background Traffic 57
5.2.1.1.1 Load of the network 58
5.2.1.2 MPEG Video Traffic 59
5.2.2 Routers 60
5.2.3 Dejitter buffers 61
5.2.4 Lost Packets 61
5.2.5 Links 62
5.2.6 Routing algorithm 62
5.2.7 Source Fragmentation 63
5.2.8 Error Propagation 64
5.2.9 Packet's attributes 64
5.2.10 Adaptation policy 64
5.2.11 Packing of RTP packets 66
5.2.11.1 Packing a complete MPEG frame in a packet 67
5.2.11.1.1 Frame loss 68
5.2.11.1.2 Average sequence delay 68
5.2.11.1.3 Delay variance of the sequence 69

5.2.11.2 Packing a slice in a packet 69
5.2.11.2.1 Packet/Frame loss 70
5.2.11.2.2 Average Delay of a sequence 70
5.2.11.2.3 Delay variance of a sequence 71
5.3 Simulation Results 72
5.3.1 Adaptation 72
5.3.2 Effects of network's parameters on PLR curves 74
5.3.2.1 Effect of the network's load 74
5.3.2.2 Effect of background traffic packet's length 76
5.3.2.3 Effect of Link Capacity 77
5.3.2.4 Effect of capacity of buffers of the routers in the PLR 80
5.3.2.5 Effect of Dejitter buffer 81
5.3.3 Comparison between RTP/UDP/IP packet sizes 82
5.3.4 Bandwidth Efficiency 86
5.3.5 Delay 90
5.3.6 Variance of delay 92
5.3.7 Increase in the maximum acceptable delay 94
5.3.8 Error propagation ratio 96

6 Conclusion 98

6.1 Summery 98
6.2 Recommendation 99
6.3 Future work 99

Table of Figures

Figure 2-1: Block Diagram of a Router 10
Figure 2-2: Fair queuing 11
Figure 2-3: Random Early Discard Function 12
Figure 2-4: Dejitter Buffer 16
Figure 2-5: Protocol Stack for real time applications 17
Figure 2-6: Weighted Random Early Discard 23
Figure 2-7: MPEG Frames Dependency 25
Figure 2-8: Intraframe Quantizer Weighting Matrix 27
Figure 2-9 : Scanning Pattern 27
Figure 2-10: Applying Prediction to make new frames 28
Figure 2-11: Bi-directional Frames 29
Figure 2-12: Model for MPEG-2 Systems in an implementation, where either of the Transport stream or the Program stream is used. 34
Figure 2-13: Putting ES in PES 35
Figure 2-14: Mapping of two PES packets into one TS packet 35
Figure 3-1: The 3 causes of degradation in a BE network 37
Figure 3-2: Syntactic error propagation in a sequence of MPEG video 41
Figure 3-3: Semantic error propagation in a sequence of MPEG video 42
Figure 3-4: Mapping of TS packets into RTP payload. 45
Figure 4-1: GoP structure 50
Figure 4-2: First stage of frame dropping 51
Figure 4-3: Second stage of frame dropping 52
Figure 4-4: Third stage of frame dropping 52
Figure 4-5: Fourth stage of frame dropping 53
Figure 5-1: The network used for simulation 57
Figure 5-2: Graph representation of the network used for the simulation 58
Figure 5-3: Simulation of aggregated Internet traffic at each link 59
Figure 5-4: The simulation's GoP structure and frames' size 61
Figure 5-5: Frame Dropping Policy 65
Figure 5-6: Chart presentation of average arrival bit rate versus PLR 67
Figure 5-7: PLR vs. Time when one slice is put in a packet 73
Figure 5-8: PLR vs. Time when one frame is put in a packet 73
Figure 5-9: Effect of network's load on PLR (Slice) 75
Figure 5-10: Effect of network's load on PLR (Frame) 75
Figure 5-11: Effect of data traffic length on PLR (Slice) 78
Figure 5-12: Effect of data traffic length on PLR (Frame) 78
Figure 5-13: Effect of link capacity on PLR (Slice) 79
Figure 5-14: Effect of link capacity on PLR (Frame) 79
Figure 5-15 : Effect of router's buffer capacity on PLR 80
Figure 5-16: Effect of Dejitter buffer capacity on PLR (Slice) 81
Figure 5-17: MPEG hierarchy in QSIF standard 83
Figure 5-18: Comparison of PLR for Frame and Slice .The six sub-plots correspond to six different network's condition as indicated on them. Dash and solid curves represent

one slice and one complete frame being encapsulated in one packet respectively. In all curves adaptation starts at the 26th seconds. 85
Figure 5-19: Headers for carrying MPEG video Elementary Stream 86
Figure 5-20: Total headers' bytes vs. MPEG frame size in SIF standard 89
Figure 5-21: Efficiency vs. MPEG video frame size for QSIF and SIF standard 89
Figure 5-22: Corresponding delay curve of Figure 5-23 90
Figure 5-23: Corresponding PLR of Figure 5-22 91
Figure 5-24: Delay curve for the case when PLR equals zero 91
Figure 5-25: Effect of network's load in delay variance for MPEG video 93
Figure 5-26: Effect of Background traffic packet length on the variance 93
Figure 5-27: Effect of maximum acceptable delay on lost packets 95
Figure 5-28: Effect of maximum acceptable delay on lost packets 95

List of Tables

Table 2: Details of the adaptation policy 66
Table 3: Effect of network's load on the probability of loss 76
Table 4: Effect of DL on the loss category 77
Table 5: Effect of increasing Link capacity to the packet loss categories (frame) 80
Table 6: Effect of Router's buffer capacity on loss categories 81
Table 7: No of fragmented packets for simulations running with different parameters . 84
Table 8: Effect of maximum acceptable delay on lost packets 96
Table 9: Error propagation ratio 97

List of Acronyms

ALF	Application Level Framing
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
BE	Best Effort
CBR	Constant Bit rate
DB	(Capacity of) Dejitter Buffer
DCT	Discrete Cosine Transform
DiffServ	Differentiated Services
DL	Data Length
DR	Data Rate
EF	Expedited Forwarding
ES	Elementary Stream
FIFO	First In First Out
GoP	Group of Pictures
ICMP	Internet Control Messaging Protocol
IP	Internet Protocol
LC	Link Capacity
MPEG	Moving Picture Expert Group
MTU	Maximum Transfer Unit
MB	Macro Block
MSE	Minimum Square Error

MTU	Maximum Transfer Unit
PES	Packetized Elementary Stream
PLR	Packet Loss Ratio
PSNR	Pick Signal to Noise Ratio
PTS	Presentation Time Stamp
QoS	Quality of Service
QSIF	Quarter Standard Input Format
RB	Router's Buffer Capacity
RED	Random Early Discard
RFC	Recommended For Comments
RSVP	ReSerVation Protocol
RTP	Real Time Protocol
SIF	Standard Input Format
SNR	Signal to Noise Ratio
TCP	Transfer Control Protocol
TOS	Type Of Service
TTL	Time To Live
TS	Transport Stream
UDP	User Datagram Protocol
VBR	Variable Bit Rate
WFQ	Weighted Fair Queuing

1 Introduction

1.1 Transmission of MPEG video over IP networks

New Internet connection technologies such as ADSL modems, cable modems and T1 lines, together with the wide spread use of the public Internet have increased the interest in developing high-quality internet/intranet based multimedia applications such as video conferencing, distance learning and telemedicine. Key to the success of such applications is the quality of received video and audio.

By introducing optical fiber to transmit data, a tremendous increase in network bandwidth has been achieved, allowing widespread use of multimedia applications. Wide area networks of global dimensions offer access to users in both the research community and businesses. Therefore network heterogeneity has become a major issue and applications have to cope with interconnected networks consisting of many sub-networks.

Computational power, bandwidth, storage and congestion control policies may unevenly vary in different network environment resulting in unavoidable congestion, delays and losses.

The transmission of images, moving or still, requires an enormous amount of bandwidth. Efficient compression algorithms have been developed but still network traffic will be easily dominated by the video. Contrary to computer data traffic, a set of mechanisms is required in order to provide good quality of service (QoS) guarantees. Circuit switching technology imposes Constant Bit Rate (CBR) utilization of the assigned channels. Thus video encoders are forced to generate a CBR stream, paying the price of varying picture quality and unsatisfying bandwidth exploitation.

Packet switched networks such as the Internet provide a basis for Variable Bit Rate (VBR) video transmission and constant picture quality. Reservation schemes like RSVP have been proposed for transmission of real time applications in IP networks. Unfortunately RSVP needs all the routers to be changed to "RSVP capable" and as we will see in the next chapter, it is very hard to be deployed in large networks. Hence, in today's "Best Effort" IP networks, VBR coding and transport introduces other potential obstacles.

Video traffic is bursty by nature. Thus, when many sources are transmitting at their peak rate the available network bandwidth may be exceeded, causing buffer overflow at the routers and packet loss. Thus, packet loss is more likely to occur in bursts, rather than be evenly distributed.

Although some loss of less important visual information may be acceptable, the introduction of compression algorithms using source coding techniques and motion compensation may lead to severe degradation in terms of picture quality when losses occur. Encoded bit streams using variable length codes (VLC) are very sensitive to losses, since the loss of a single bit forces the decoder to discard information until the following synchronization sequence.

Several informational video compression standards exist, MPEG1/2, H.261/3, Motion-JPEG as well as several proprietary streaming video solutions such as Real Video,...

The Moving Pictures Expert Group (MPEG) has developed a widely used standard for video compression. The standard deals with two kinds of frames: Reference frames (intra frames) carry the complete set of parameters needed for frame reconstruction at the decoder, whereas other frames (inter frames) contain only information about changes from these reference frames.

TCP, the reliable transmission protocol used in internet community, can not be used for transmission of real time applications due to unacceptable delays that it imposes, so instead User Datagram Protocol (UDP), which is unreliable, coupled with Real Time Protocol (RTP) is used to for transmission of video over IP networks.

1.2 Problem Statement

Due to the dependency of information in the MPEG video and because of unreliability of UDP, transmission of MPEG video over IP networks is much more sensitive to loss and delay than other kinds of information. Any error in an MPEG frame is propagated to the other frames, and the effect of the error depends on the location of it. Also for real time applications, delay cannot be tolerated more than a certain level.

To address this problem, one solution is trying to control the degradation, so that instead of an unpredictable degradation, the receiver receives the video sequence that although is not as it was when encoded, but the degradation has been somehow controlled to enhance QoS. One of these ways is adaptation, which will be discussed in detail in chapter 3 later. Various adaptation techniques could be deployed for controlling the degradation and improving QoS. This thesis considers a source driven adaptation technique for delivering of MPEG video over IP networks and examines its effectiveness through a simulation and then investigate the effect of different network parameters on the technique. Furthermore, since (as will be described in detail later) the standard for packetization of RTP/UDP/IP packet does not detail about size of packets, the research proposes two packetization schemes and compares these two and studies their effects on the QoS of the receiving video.

1.3 Organization of the Thesis

This thesis is organized as follows:

Chaptertwocovers the background material related to the work presented. This chapter is divided totwosections: The first section covers IP networks with emphasize on those aspects of best effort networks related to real time transmission of video over them. Section two covers basics of MPEG video.

Chapter 3 covers some topics on transmission of MPEG video over IP networks that will be used during the remaining part of the thesis.

Chapter 4 introduces the main idea behind this thesis. A source driven adaptation scheme is proposed. Then two different packetization schemes for this adaptation policy will be described.

Chapter 5 describes the simulation and its results. Definition of different parameters is given and implementation of the schemes explained in previous chapter is described. Results of the simulation are given and interpreted.

Chapter 6 concludes the thesis by summarizing what we have done and proposes some work to extend this subject in future.

2 A quick look at IP networks and MPEG video

This chapter introduces the basic theoretical background for the thesis. In part one best effort networks and real time applications over them are described. In part two basics of MPEG video are described.

2.1 Part one

2.1.1 Real time applications and IP networks

In part one we briefly introduce IP networks and their most important characteristics. TCP/IP protocols are thousands of pages and neither it is the intention of this chapter nor is it possible to describe them here. We only highlight those aspects of IP networks *that are directly related to real time applications* and briefly introduce the techniques suggested to improve the best effort model of IP networks to a model suitable for such applications.

2.1.1.1 Application-level and Network-level interconnection

Packet-switched networks has been organized and spread all around the world during a relatively long period of time. With existence of many individual "local" and "wide" networks with different technologies (Ethernet, FDDI, X.25, ATM,...), the logical question that arises is how can we connect these individual networks to each other so that one host in a network to be able to communicate with another host in another network build based on a completely different technology?

Two approaches could be made to address this problem: Application level Interconnection and network-level interconnection or *internetworking*. The former tries to provide uniformity through application level programs called *application gateways* running in each computer attached to the network. The computers are able to understand the details of the networks connections and inter-operate across those connections with application programs on other computers. The difficulty with this approach is that

adding a new functionality to the network needs adding new programs to all the computers and adding new hardware necessitates modifying or changing the programs running in all the computers. Internetworking on the other hand breaks all the messages between the
communicating computers to small packets of data and deals with these packets without using intermediate application programs. The benefits are several: breaking messages to small packets not only enables them to map to the underlying hardware hence improving efficiency, but also permits the intermediate hardware handle network traffic without the necessity to understand the application programs which send and receive them. Furthermore, this approach makes the whole system flexible allowing adding or changing network without changing the application programs. These benefits have caused excessive usage of connection of networks through internetworking. The biggest one, the global Internet today connects tens of millions of computers with a growing rate of 100% each year. Internet Protocol or IP takes care of network-level interconnection of networks.

2.1.1.2 Internet Protocol (IP) [1]

The Internet Protocol (IP) is the key tool today to build scalable, heterogeneous internetworks. To be successful, the two most important issues that should be considered by IP are *heterogeneity* and *scalability*. Heterogeneity is the ability of interconnecting of networks with any kind of technology, (even those that have not been invented yet), and scalability is the ability to manage the growth of the network of networks to any extent.
To address heterogeneity, IP hides all the "messy" details of the individual underlying networks from the upper protocols or applications, so that when working with an "internetwork" (or internet for summery), the upper levels do not need to know anything about the underlying technologies. IP resides above each particular network and isolates the particular of each network. The upper protocol or application only delivers the data to IP and it is the responsibility of IP to transmit the data across real networks.
To achieve the scalability, the amount of information that is stored in each router and that is exchanged between routers should be kept as low as possible. IP assigns a unique and

global address to each host connected to the internet. The addressing is assigned hierarchically and should have the ability to be reconfigured, so as if the network configuration changes (adding or removing hosts or networks), new addresses could be assigned. IP addresses are comprised of two parts: address of the network that the host connects to, and the address of the host in this network (link-level address). The nodes just keep track of the network addresses, i.e.; they only contain forwarding tables that list only a set of network numbers, rather than all the hosts in the network. This helps to maintain the forwarding tables as short as possible.

A protocol called Address Resolution Protocol (ARP) enables each host in a network to build up a table of mapping between IP address and link-level address, hence enabling the exchange of packets between the hosts having the same internet address. [2]

IP is a connectionless "best effort" packet switching protocol. All the data given to IP are put inside (possibly variable length) packets of "datagram" and are sent to the destination with no guarantee to be delivered.

Datagrams include a payload of data which could be up to 64KB length, together with a header containing useful information such as: source address, destination address, a field called Time TO Live (TTL) which is either a timer that as it reaches zero, the datagram is discarded, or is a counter which is set to a certain value (64 is the default value) at the beginning and is decremented as datagram passes through routers. TTL is used to prevent the packets that have not been able to reach the destination or those packets which are trapped in the loops to become cleared from the network after their TTL reaches zero and they are not delivered yet. Another field which has not been used so far, but with the introduction of the Differentiated services (Diff-Serv), has become very important is an 8 bit field called TOS (Type Of Service) which was supposed to specify how the datagrams should be handled at the routers. The first 3 bits of TOS known as *precedence* indicate the importance of the packet and the next 3 bits indicate if the datagram needs low delay, high throughput or high reliability. The two other bits are unused. What was done in practice was that in most of the cases only precedence value of 011 and 111is interpreted as network management data and it is treated more carefully. [3]

A companion protocol called Internet Control Message Protocol (ICMP) reports a collection of error messages to the sending host if a datagram fails to be delivered to the destination for any reason or a syntactic or semantic error in the IP header is noticed. The messages are sent from hosts and routers to the sending host and are encapsulated in IP packets, hence they themselves are likely to be lost.

Because IP is connectionless and since there is no "virtual circuit" through which the packets are sent, it is probable that those individual packets that reach the destination has traversed different routs with different delays, which may result they reach the destination out of order.

2.1.1.2.1 Fragmentation and Re-assembly

As we described before, IP is a protocol for connection of different networks of different technologies. Any networking technology, Ethernet for example, among the other characteristics like having its own medium access scheme, uses its own frame format which its maximum frame size (1500 byte in the case of Ethernet). When IP packets are generated and forwarded to the destination, they pass through different networks. In order for IP packets to be able to be transmitted over different networks, their size should not exceed the maximum allowable size of that network .two approaches can solve this problem. One is not allowing the source to generate IP packets that exceed the minimum size of the largest possible frame of the technologies that make the internet. For example if the underlying technologies are FDDI, Ethernet and PPP with maximum allowable frame size of 4500, 1500 and 512 bytes respectively and at the source generate IP packets that have the size of 512 bytes or less, we will not be faced to any problem. This has the disadvantage that if a packet does not even need to pass through PPP links, it has been forced to be broken to several packets with their own headers, which are to be processed one by one by the routers and obviously this decreases the efficiency. The other approach is to let the IP packets to be sent at the maximum allowable size of the network which the source is connected, and then if during their transmission, they have to be passed through a network with smaller maximum frame size, they may be "fragmented" to several IP packets with the same identity and continue their way to the destination. At the destination, the host "reassembles" the fragmented packets to the original IP packet and delivers it to the upper layer. The drawback here is that if one fragment of a packet does not reach the destination, the whole packet will be considered lost so the probability of losing a packet increases. That is why a process known as "path MTU discovery" is under serious consideration. In this process fragmentation is avoided by sending packets that are small enough to traverse the link with the smallest allowable frame size in the path from sender to destination. *In our simulation we use this process, hence the fragmentation is done at the source hosts.* This is consistent with the new IPV6 protocol too.

2.1.1.2.2 Delay of Datagrams

The delay of a datagram is the time it takes for a datagram from the moment it is generated (i.e., encapsulated in an IP packet) till the moment it is delivered to the upper protocol at the end host. This delay is the sum of packetization delay, Link delay, queuing delay and processing delay at the routers.

-Packetization delay is the time that the first bit of a datagram experiences as it arrives inside the IP packet till this packet is forwarded from the source. For delay sensitive applications, the size of a packet cannot be large, as this will cause unacceptable delays. For example if we encapsulate every 500 bytes of a 64 Kb/Sec PCM voice into a datagram, the first bit of the datagram should wait:

$$500*8/64000=63 \text{ ms}$$

until the packet is ready for dispatch, a clearly long time for IP telephony.

-Link delay is the time that the first bit of a datagram enters a link till the time that the last bit of this datagram exits the link. Link delay is affected by the propagation delay, which in turn is a function of the length of the link and the capacity of the link. For a certain packet, it takes more time to pass through say 1000km length copper link with capacity of 1Mb/Sec comparing to the same link if the capacity is increased to 2Mb/Sec, while the propagation delay is the same for both cases.

-Queuing delay is the time a packet should wait in the queue or queues (input and output queues) of the IP switches (routers) in order it get access to the link.

-Processing delay is the time that it takes for a datagram's header to be processed by the router and to be forwarded to the correct output. The processing normally is done by the input ports as described in the next section.
To these delays, the delay resulted from fragmentation and reassembly should also be added.

2.1.1.3 Routers [4]

IP connects individual networks to each other by "routers". Routers are devices that receive IP packets from the link connected to one out of its m inputs and forward it to one or more of its n outputs. Such a router is called an m*n router. Obviously if bi-directional links are connected to the router, then m=n.
Routers are often characterized by a packet per second rate as well as a throughput in bits per second, when the former one is measured with the minimum-sized packets. Routers consist of three parts: *Ports*, *fabrics* and *port mapper*. Ports communicate with the outside world. They may consist of receivers, buffers that keep those packets that are waiting to be routed or transmitted. The only task of the fabric is to forward the IP packet it receives to the relevant output port. To do this, normally it is the input port that figures out where the packet should go. Then it directs the fabric to the correct output port either by adding the port no. to the packet (self-routing) or by sending relevant control instructions to the fabric. Routers may have buffers at input ports, output ports or fabric (internal buffering). If the buffers are at the input ports, it is probable that two or more input buffer want to send a packet to the same output port at the same time. In this case only one packet can be forwarded to the output and the others should wait, making the packets behind them not to be processed, although there may be no contention for their relevant outputs. This phenomenon, known as *head-of-time-blocking*, decreases the throughput of the router. That is why the majority of the routers use either pure output buffering or a mixture of internal and output buffering. In our research and during the simulation, the routers are assumed to be pure output buffered routers.

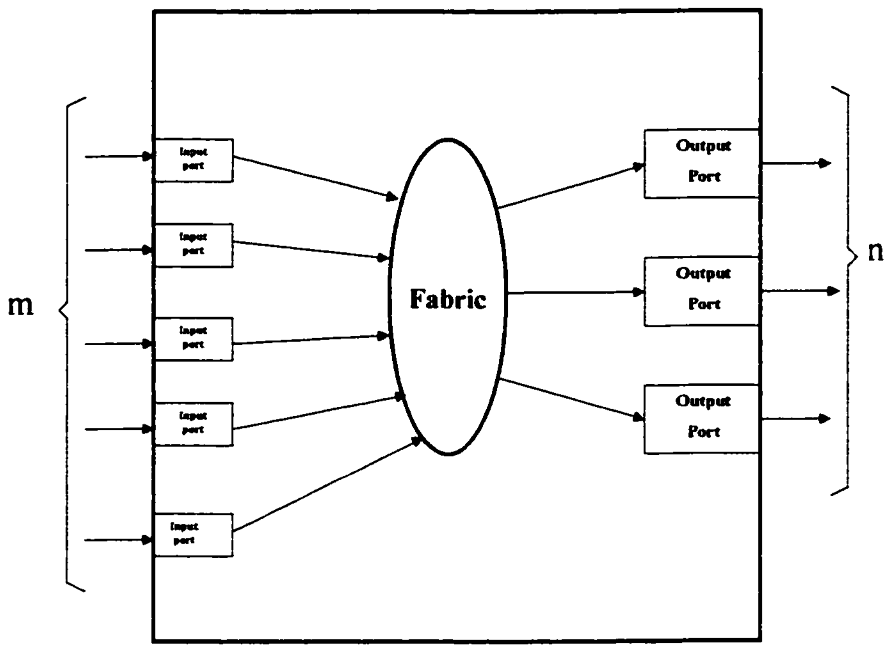


Figure 2-1: Block Diagram of a Router

2.1.1.3.1 Queuing in the routers

Most of the routers in the current Internet use FIFO (First In First Out) as their *scheduling discipline* and "tail drop" as their *dropping policy.* In a FIFO the packets are served (transmitted in the case of pure output buffering) based on the order they have arrived the buffer and tail dropping means that in case the buffer is full, packets that arrive at the end of buffer are dropped. FIFO queuing is widely used in the routers and the fastest available routers, as far as packet per second is concerned, use FIFO queuing.

Fair queuing is another queuing algorithm that is used to address the inability of FIFO to differentiate among flows to ensure equal access to service. Fair queuing tries to keep "greedy" applications from grabbing more than their share of the bandwidth. The mechanism is that it isolates traffic flows from each other and maintains a separate queue for each flow currently being handled by the router. The router then services the queues in a round-robin manner as shown in Figure 2-2

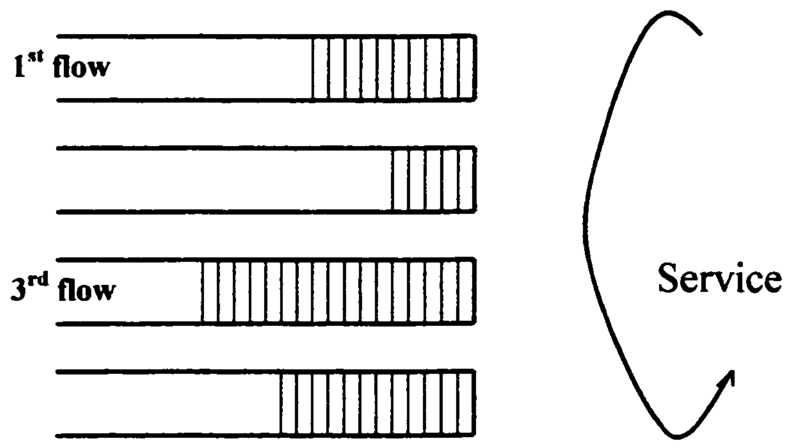


Figure 2-2: Fair queuing

Since the packets in each flow possibly have different lengths, to maintain fairness, the actual service given to each flow is more complicated than above so that a kind of bit-by-bit round ribbon service is simulated considering the length of the packets.

*Weighted fair queu*ing (WFQ) has been designed to improve performance of small-packet flows, such as speech, compared to high-volume, large-packet flows such as file transfer. WFQ gives low volume traffic priority over high-volume traffic, which helps move these small packets through the system rapidly. WFQ does this by sorting and interleaving individual packets by flow and queuing each flow based on the volume of traffic in each flow, thus preventing larger flows (i.e., those with grater byte quantity) from consuming bandwidth. This is the *fairness* aspect of WFQ-ensuring the larger traffic flows do not arbitrarily starve smaller flows. The *weighted* aspect of WFQ is done by applying IP precedence bit in TOS field of the IP packets.

Priority queuing [5] is another method of queuing used in the routers. We can imagine priority queuing like fair queuing, in which each flow is forwarded to a different output FIFO, but as long as there is one packet in the FIFO with the highest priority, service is not being given to the other queues. So in this case an stream which needs a bandwidth less or equal the bandwidth of the link connected to the output of the router could be guaranteed to receive it, of course in the expense of blocking the other flows from the router's service. In practice, there is only one FIFO and packets with the highest priority are transmitted ahead of the other packets, so that this traffic is always sent ahead of other types of traffics. This needs that the router's processor spends more time to check all packets to insert the arriving packets in the correct position in the queue.

Note that FIFO scheduling can use another approach like Random Early Discard (RED) as its dropping policy. Since a sort of RED is used in Differentiated Services (Diff-Serv) described later, we below give a brief description of it now.

2.1.1.3.2 Random Early Discard (RED) [6]

RED assigns two threshold values to a FIFO: Tmin and Tmax. If queue's length is more than Tmax, tail dropping is applied. If the queue's length is less than Tmin, the new packet will be accepted and will be buffered at the tail of the queue. If the buffer contains packets between Tmin and Tmax, then the newly arrived packet is discarded with the probability of p. P linearly increases from 0 to 1 as the queue's length moves from Tmin to Tmax. (Figure 2-3)

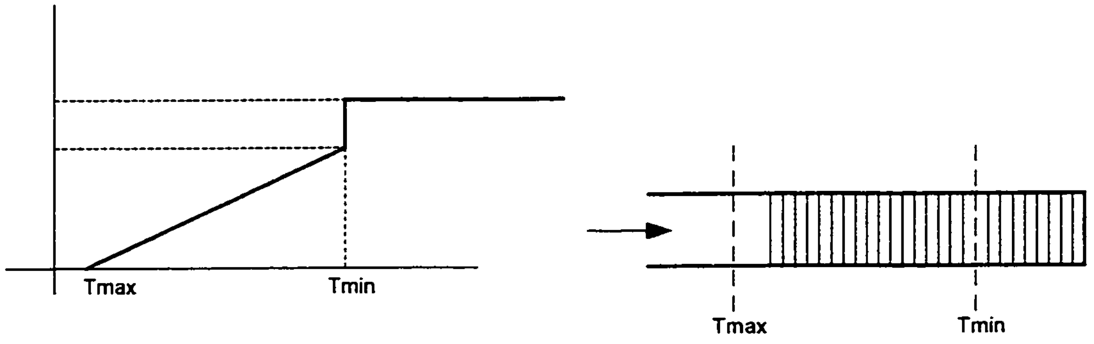


Figure 2-3: Random Early Discard Function

Since the IP packets have a bursty nature, it would be misleading if we judge about a queue by just looking at its instantaneous length. So instead of comparing instantaneous values of queue's length with Tmax and Tmin, a sort of averaging is done for bypassing the sudden changes in the length of the queue. It is like applying a low-pass filtering to the queue's length.

Average length is calculated as below:

$$\text{AvgLength} = (1-\alpha)\ \text{Old\_AvgLength} + \alpha * \text{Current\_queue\_length}$$

α is a weighting factor and $0 < \alpha < 1$ with typical value of .002.

RED was mainly designed as a dropping policy used by TCP for congestion avoidance. After implementation of in the Internet it was observed that RED tends to drop a cluster of packets instead of dropping them distributed in time. When a group of packets are dropped together, because normally they belong to the same sender, this will cause that this sender reduces its rate too much while the other users who are also responsible for congestion do not decrease their rate. To solve this problem of "unfairness", means were provided in order the probability of dropping not only depends on the average length of the queue, but it also depends on how long it has been since the last packet was dropped. In the other words, if the last packet was dropped a long time ago, the probability of dropping of a newly arrived packet is much more than if the last packet was dropped just before arrival of the new packet. In this case the probability of dropping packets is distributed in time and router tends to drop packets spaced in time. To achieve this goal we define:

$$\lambda = \text{Max p} * (\text{AvgLength}-\text{Tmin})/(\text{Tmax}-\text{Tmin})$$

and p, the probability of dropping a packet is calculated as:

$$p = \lambda/(1-\text{count} * \lambda)$$

count keep track of how many newly arriving packets have been queued (not dropped) while AvgLength has been between the two thresholds. As the time between dropping of the two packets increases, p also increases, make things less probable for the packets that the packets before them are dropped. This property gives RED a kind of "randomness" in treating the packets, which allows discarding packets of different flows, which is very desirable to prevent global synchronization in congestion avoidance mechanism of TCP. In TCP, receipt of a packet is acknowledged by the receiver. Any packet which does not receive an acknowledgement is considered to be lost. Loss of a packet, which is mainly due to congestion, causes TCP decrease the rate of transmitting by reducing its "window-size" by half. Now if a group of packets are discarded together in a router, they may belong to several users. In this case, TCP will react and orders all of these users to considerably reduce their window-size, while this amount of reduction is not necessary to solve the congestion. This phenomena known as "global-synchronization" is not desirable and should be avoided by dropping packets randomly from different flows.

Note that although TCP provides a "Congestion Control" mechanism, which means it reacts after congestion happens, RED is a "Congestion Avoidance" mechanism which tries to prevent congestion when it is likely to happen.

2.1.1.4 User Datagram Protocol (UDP) [7]

It is possible that several application programs in a host exchange information with several applications at other host(s) simultaneously. IP delivers the datagrams to each *destination host* through ARP without considering that to which of the running processes on this host these packets belong. To take care of that the datagrams are delivered to the right process, a sort of "demultiplexing" is necessary which direct the packets to the appropriate process. UDP takes care of this task by assigning a *port* to each application. A port is an address where an application makes itself available on a particular host and is typically made by a queue. As the UDP datagrams arrive the host, they are put in their relevant queue and are served by the relevant application. A "checksum field", checks the correctness of the packet and if it is not received correctly, it is discarded. If the queue is full, UDP datagrams are simply discarded. No other additional functionality, such as informing the source of any discard or out of ordering receipt of datagrams, is added to IP by UDP.

2.1.2 Real time applications over IP

Some applications over IP networks need to receive the transmitted IP packets within a limited amount of time after they have been transmitted. We call these applications as "real time " applications. For example, it is well known in the communication industry that if in a telephone call, the receiver does not hear the sender's voice after 300 ms, the quality of that voice will be poor for telephony. These real time applications may or may not tolerate data loss, i.e., although they are sensitive to delay, but a subset of them can work fine even if some packets are lost, but other group of these applications need that not only the data to be received on time, but it should arrive completely as it was sent. For example although in Internet telephony applications face degradation with loss of some packets, but the quality remain acceptable if data loss does not increase a certain level. In the case of controlling a robot through sending the instructions by IP packets, on the other hand, time constraint is not enough for guaranteeing its correct functionality, but the instructions should receive without any error. We call the first group "loss tolerant". Sometimes tolerance is defined for variation in delay or jitter too. In this case, tolerant application are those that can tolerate jitter to some certain level, like various audio or video streaming. Examples of intolerant to jitter applications are two-way telephony applications.

A real time application may or may not be "adaptable". By adaptability we mean that according to the condition of the network, we can change one or several parameter of the application so that it still works "fine" (with certain definition of fine) in the new conditions. Consider the Internet telephony again and assume that average delay of the packets is increased from 150 ms to 300 ms. If we manage to control the play back time of the packets by adjusting the play back time so that under the new conditions they are played back after 300 ms after generation, we have adapted the application to the new conditions in the network so that it still works "fine". In case the delay increases, there is no room for further adaptation as the quality will not be acceptable any more. As another example, in streaming a video clip over a network, if we manage to reduce the load of the network by injecting fewer bits while keeping the quality of it acceptable, we are dealing with an adaptable real time application. Real-time applications usually sit on top of UDP.

2.1.2.1 Real-Time Transport Protocol (RTP)

Multimedia applications have common requirements. One of these requirements is timely playback of them. When a video or voice clip is sampled at a certain period of time and put inside a datagram, at the destination it should be played back so that its time interval with the previous datagram to be equal to the time interval of the two datagrams when they were sampled at the sender side. In other words, not only the delay that each packet tolerates when it passes through source to the destination is limited, but also this tolerable delay should be kept constant in order we do not experience variations in the delay of datagrams which will decrease the perceived quality. Variation in delay is called *delay jitter.* Delay jitter could be avoided by buffering a certain amount of datagrams before we start playing them back in a so-called "dejitter buffer". This helps to have a smooth playback of the clip. (Figure 2-4) In order each datagram knows when it should be played back relative to its previous datagram, a sort of "time stamping" should be applied to each datagram.

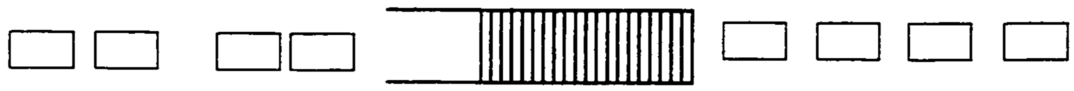


Figure 2-4: Dejitter Buffer

Another requirement of multimedia applications is to provide means in order the receiver have an idea of how many and which datagram are lost. The importance of "how many" is that the application can get an idea of the congestion condition of the network in order to ask the sender to adapt its rate to new conditions of the network. Sequencing of the datagrams enables to achieve this goal.

The importance of "which" is that sometimes applications should behave differently with the kind of datagram that has been lost. For example if a datagram that includes header of an I frame of an MPEG video clip is lost, the application behaves differently with when a datagram of a B frame is lost. In the former case, the application should somehow conceal an error that propagates within the whole Group of Picture, which is obviously is different from concealment techniques used when the lost datagram belongs to a B frame. Marking the frame's boundaries helps to identify different frames (MPEG frames in above example).

The requirements mentioned above has led to specification of Real-time Transport Protocol (RTP) [8], which in addition to time stamping, sequencing and marking also provides some other important functionality like providing information to enable to synchronize multiple media (video and voice for example). RTP provides unreliable, end-to-end delivery services and normally is used above UDP to make use of its multiplexing and error-indication services. Figure 2-5 shows the protocol stack for real time applications. There are many different multimedia applications, each with their different requirements and these requirements are best understood by the application themselves. That is why RTP just specifies certain fields, which are likely to be used by many different applications, in its header and leaves many of the protocol details to the application. Using properties of the payload in design of data transmission protocols is called *Application Level Framing* (ALF). The idea behind ALF is that the applications should be involved in the data transmission process, because applications know the requirements of the information being transmitted. This information should be packetized into application data unit (ADU) bits. It is the application then that determines if ADU's should be retransmitted, discarded, or reordered.

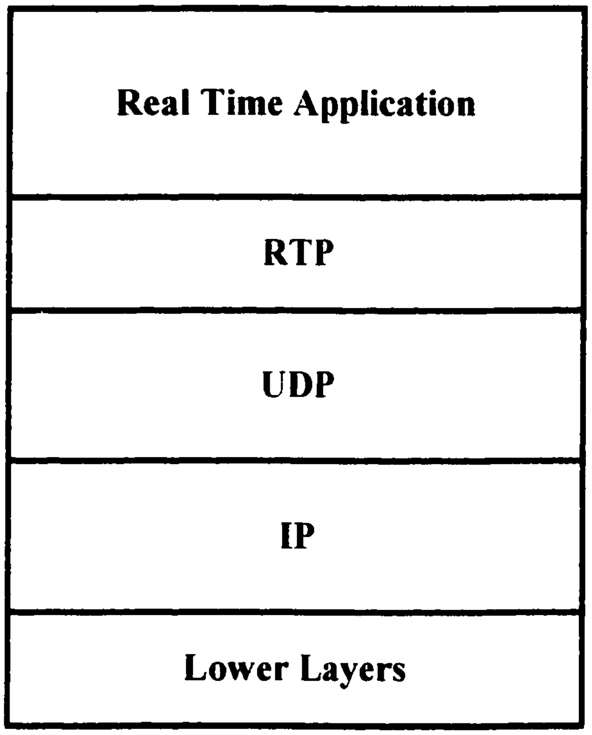


Figure 2-5: Protocol Stack for real time applications

2.1.2.2 Quality of Service (QoS) [9]

Quality of Service is a vague term often referring to the methods for classifying network traffic and ensuring that some of these traffics receive special handling. The "special handling" is application dependent and could range from keeping the latency, delay jitter, error rates, ... beyond a certain level. A network that could provide such different levels of services is told that it supports QoS. Obviously the best effort service model as discussed so far, is not able to provide such guarantees. The only QoS provided in an IP header is TOS byte, but majority of TCP/IP protocol stacks did not allow users to set that byte through APIs. So one is not able to change QoS level after a network administrator has assigned it. Even if the network administrator wants to provide certain QoS levels to some users, he/she should predict all of the possible paths for each flow, calculate the buffering sizes for the flows, consider latency of links and queuing delays for each flow and all flows that have requested such services as a whole, ... and then begin to manually configure the routers, a process which obviously is very difficult and time-consuming and maybe impossible for a large network. The situation becomes more difficult if a new user is added to the network, making a new path, requiring reconfiguration of the routers in that path.

To address these problems, means should be provided in order the best effort service model to be improved to support QoS.

Basically two approaches could be made in order to provide QoS to IP networks:

-*Fine-grained* approaches, which provide QoS to individual applications or *flows*.

-*Coarse-grained* approaches which provide QoS to large classes of data or aggregated traffic.

The first group, often referred as "guaranteed services", includes RSVP (Resource Reservation Protocol) as its most important technique. In the second group we have more recent approaches known as Differentiated Services or Diff-Serv.

2.1.2.3 Integrated Services (RSVP) [10]

RSVP is a completely new protocol with its own protocol ID. RSVP packets are carried by IP, therefore using the same routing tables and follow the same routs as IP packets. The important difference is that RSVP packets interact with the routers when passing through them; hence any change in the routing tables, as it happens occasionally, is reflected in RSVP packets too.

In RSVP, the aim is that the necessary resources for a "flow" of packets to be allocated to it, in order to guarantee a certain QoS for it. A flow is defined as a sequence of data packets sharing the same combination of source and destination address, along with any other distinguishing characteristics that maybe differentiate it from the flows sharing the same address pair. In some cases, the identifying information for a flow is referred to 5-tuple (the combination of the source and destination addresses, IP port numbers and protocol type).

To begin the process of resource reservation, the source sends a special message called "PATH message" to the destination. As the messages passes through "RSVP-aware" routers to reach the destination, it sets their control blocks so that they will be able to recognize the corresponding flow when it appears at one of their interfaces.

PATH message also give some important information about the characteristics of the flows like average and peak data rates, etc to the routers. These information is called "Traffic Specification" or *TSpec* and is normally given is the form of a *token bucket* filter parameters. As the "PATH message" passes through routers, they also set some of its fields. These fields indicate the QoS services the routers can provide and how much resources they can afford to dedicate to this flow. Hence when the PATH message reaches the destination, not only it has introduced the flow that wants to use RSVP to the routers, but also has gathered valuable information about the path that the flow will take. Note that so far no reservation has been made. *It is the destination that requests the reservation.* The reason is that in multicast cases, where a source sends many flows to many receivers, it makes more sense that instead of the sender keep tracks of the reservation for all receivers, each of the receiver itself take care of its necessities.

The reservation is made through RESV message, sending from source to destination and passing through exactly the same route as PATH message passed through. This message includes TSpec and "Requested Specification" or Spec, which describes the requirement of the receiver for that flow. Each router looks at the reservation request and tries to allocate the necessary resources to satisfy it. If this RESV message is able to reserve the necessary resources through all path, it will send the destination a confirmation signal and the transmission begins, other wise a fail signal is transmitted by any router that has not been able to allocate the requested resources.

One important characteristic of connectionless protocols like IP is their robustness against any failure that may happen in a part of the network. To remain faithful to this robustness, RSVP reservation method, as described above, should be able to change the reservation through different paths if necessary. That is why the PATH and RESV messages are sent *periodically* between source and destination in order to address the dynamic nature of IP routing paths. In fact, when PATH and RESV messages refresh the routers control blocks, they also reset an associated timer indicating how long the control blocks should keep that setting. If a router is not refreshed by a new PATH and RESV message, this indicates that this router is no more in the flow's path, hence there is no need for it and, as the timer reaches zero, the RSVP control block is cleared. This technique is known as *soft state.*

RSVP is able to make two basic reservation classes: *Guaranteed Services*, which provide a bandwidth guarantee and a maximum end-to-end delay, and *Controlled Load*, which provides a priotorized service that limit the total amount of controlled load traffic on a link such that the load is kept reasonably low.

Which of the above services is selected depends on the application and the receiver's decision. For tolerant real-time applications, Integrated Services model recommends the use of controlled-load services and for intolerant real-time applications guaranteed services are recommended.

The *Controlled Load Service* actually provides "better than best effort" delivery. The controlled-load service does not accept or use specific values for control parameters that include information about delay or loss. However, the degree at which the traffic maybe dropped or delayed should be less enough in order the real-time applications to function without noticeable degradation. For example the router should be prepared to occasionally provide more than specified in the flow's Tspec, in order to prevent queue buildup made by probable presence of a bursty traffic at the router or if the flow send some traffic which are not conformant with what was specified in its Tspec, while the

router provides the necessary amount of resources to the conformant traffic, it should manage to provide best effort service to the nonconformant traffic too.
The guaranteed services on the other hand guarantee that if the packets remain within the specified Tspec, they will arrive within a certain delivery time and will not be discarded because of buffer's overflow at the routers. The only promise that this service gives is about the maximum queuing delay. It does not guarantee anything about minimum or maximum (end to end) delay or variation in the delay (Jitter).
It is obvious that a FIFO queue in a router is no more suitable for these services, and specially guaranteed services need more complicated queuing, like a weighted share queuing discipline in which each flow gets its own individual queue with a certain share of link.

2.1.2.4 Differentiated Services [11]

The ability of RSVP to be made in large networks is still under question. This arises from the fact that remembering each flow's and its state by the RSVP-aware routers, as the networks grows becomes more and more an overwhelming task.
Differentiated classes of services although does not provide granularity of RSVP, but is much simpler to be managed. Here a router does not have to remember a "flow" and its relatively detailed individual specification and requirements (Tspec, RSpec), but it tries to give different levels of priorities to the *individual* packets as they pass through it. The idea behind Diff-Serv is very simple: If we manage to put the information about the special service that any packet needs inside its header, the router will be able to recognize it and give the appropriate service to this packet. The "appropriate services" are called Per Hop Behaviors (PHB) and can be defined in many ways and in fact they are still under consideration by the IETF.
The disadvantage of treating individual packets is that here the flows are treated as a part of large groups of packets without being able to single out any individual flow for special handling. On the other hand there are many advantages compared to this disadvantage that make Diff-Serv very interesting for network administrators.

One of the advantages of Diff-Serv is the ability of IPV4, which is the dominant version of IP currently used, to use this service. The field "TOS" at the header of IPV4, has never been used widely as it was intended. Hence this byte could be used as a "Differentiated Services" or DS-byte, or even if it is used as it was defined previously, the routers at the edge of DS enabled network will over write it by the appropriate information.
6 bits of DS-byte is used for identifying the PHB's, which should be applied to PHB's. Diff_serv introduces two classes of services for PHB's: Expedited forwarding (EF) and Assured Forwarding (AF).
Expedited Forwarding tries to provide the assured bandwidth, low-latency, low-jitter, low-loss behavior to the packets. Exactly how the routers provide such service could be different and is left to network administrators. They may give absolute priority to such packets, which of course could lead to locking out of essential routing traffics. In this case any EF packets interrupts the router in order to receive the service. One may give less higher priority to such packets which prevent the important network management packets to be locked but on the other hand may cause that some of the EF packets does not receive the service they require. This could be done by WFQ and assigning much higher weight to EF packets queue, but also giving a little weight to other packets. One

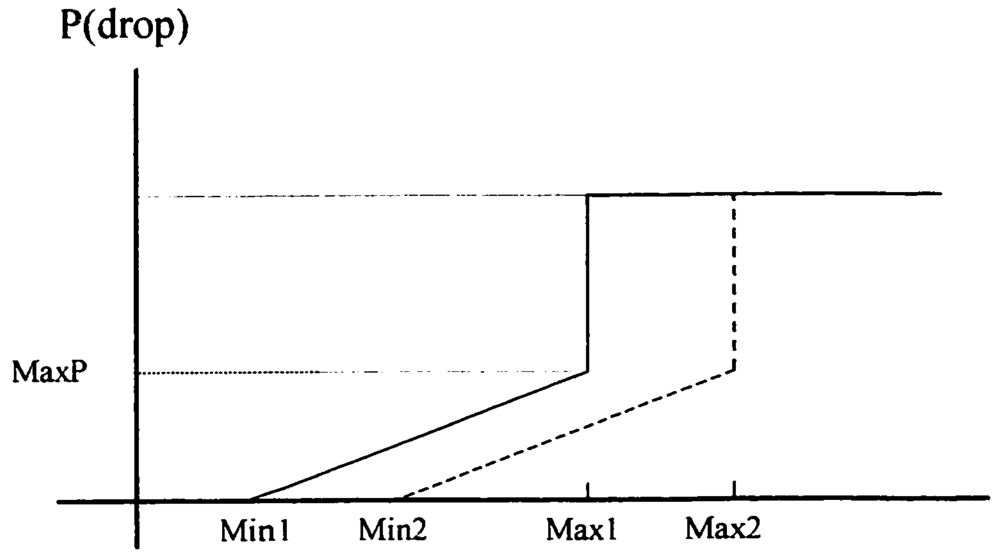


Figure 2-6: Weighted Random Early Discard

conservative approach is not allowing the total EF input traffic at the edge of a network to exceed the bandwidth of the slowest link in the network.

Assured Forwarding on the other hand combines prioritization with packet drop probabilities. This could be done by assigning two or more Tmin and Tmax to a RED buffer as shown in the Figure 2-6. Here as the buffer start to grow, the probability of dropping of the packets with less priority increases faster than the probability of dropping packets with higher priority, so that all the packets of lower priorities will be dropped if the queue length increases Tmax of this group (Max1 in Figure 2-6).

2.2 Part two

2.2.1 A quick look at MPEG video

MPEG is a series of standards established by **Motion Picture Expert Group** of International Standard Organization (ISO) for audio/video compression. So far three versions of MPEG standards (MPEG-1, MPEG-2 and MPEG-4) has been officially released while the 4th and 5th series (MPEG-7 and MPEG-21) are under development.

MPEG-1 was primarily intended to be used for stored, error free environment applications like CD-ROM and it targeted approximately 1.2 Mbps. MPEG-2, was designed for wider applications. It supported interlaced video, larger screen formats, hence making it suitable for TV video applications. MPEG-4, officially released in 1999, supports very low bit rate audio-video communications that makes it suitable for wireless multimedia.

2.2.1.1 Video MPEG [12], [13], [14]

We concentrate on *video* part of MPEG-1 and MPEG-2 (both hereinafter called as only "MPEG" unless otherwise stated), as the other parts are irrelevant to this research.

MPEG takes advantage of all the advanced compression techniques known today. It uses:

-Frequency domain decomposition which uses DCT (Discrete Cosine Transform) to exploit statistical and perceptual spatial redundancy.

-Temporal prediction: to exploit redundancy between video pictures.

-Quantization: selective reduction in precision to reduce bit rate while minimizing loss of perceptual quality

-Variable-length coding to exploit statistical redundancy in the symbol sequence resulting from quantization.

2.2.1.2 MPEG Frames

MPEG generates 3 kind of frames namely I (Intraframe), P (predictive) and B (Bi-directional). I frames are standalone frames which are generated independently of other frames, providing random access to a sequence of video. P frames are generated according to the previous I or P frame by applying motion estimation to a group of pixels and B frames are generated based on applying prediction to previous P (or I) and next P (or I) frames. Hence P (and B) frames are dependent to their previous (and next) frames(see Figure 2-7). I and P frames are called "anchor frames" in MPEG terminology.

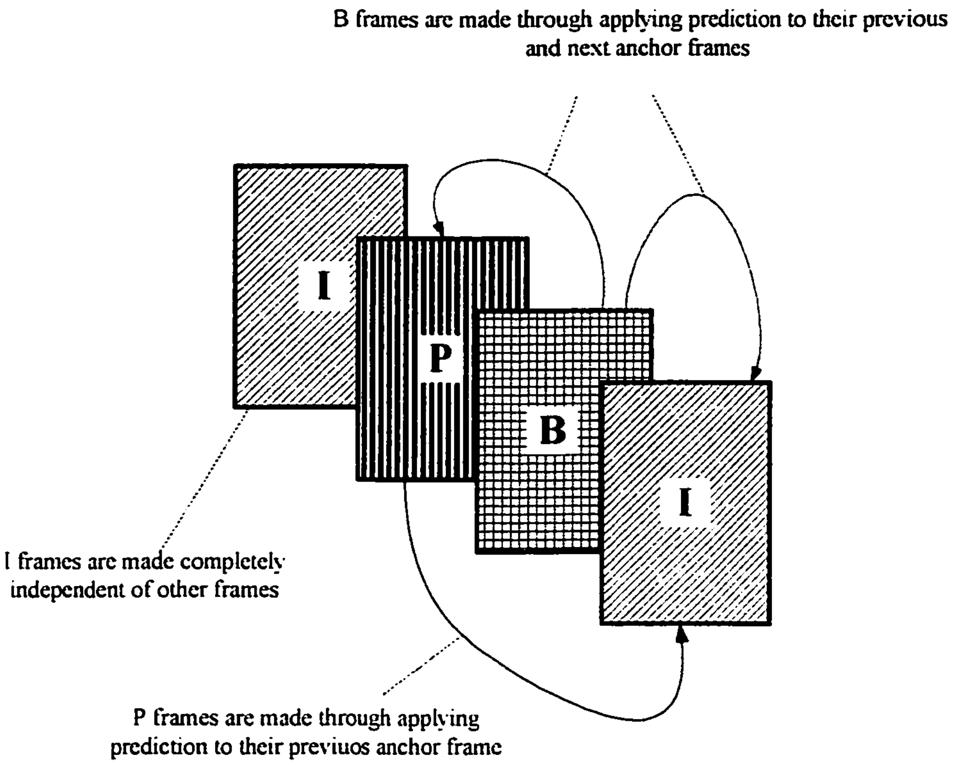

B frames are made through applying prediction to their previous and next anchor frames
I
P
B
I
I frames are made completely independent of other frames
P frames are made through applying prediction to their previuos anchor frame

Figure 2-7: MPEG Frames Dependency

2.2.1.3 Intra-frame encoding

To make a compressed I frame from the input frame, it is divided into regions of 8*8 pixels called blocks. To each of these blocks DCT is applied.

DCT is defined as:

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} pixel(x,y) \cos[\frac{(2x+1)i\pi}{2N}] \cos[\frac{(2y+1)j\pi}{2N}]$$

$$pixel(x,y)=\frac{1}{\sqrt{2N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}C(i)C(j)DCT(i,j)\cos[\frac{(2x+1)i\pi}{2N}]\cos[\frac{(2x+1)j\pi}{2N}]$$

$C(x)=\frac{1}{\sqrt{2}}$ if $x=0$ and 1 if $x>0$.

Where $pixel(x,y)$ is the grayscale value of the pixel at the position (x,y) in the 8*8 block being compressed and N=8 in this case.

What DCT does is that it translates the 8 by 8 input matrix of pixels to an 8 by 8 matrix of frequency coefficients so that the finer details of the input which correspond to higher spatial frequency to be separated from grosser details which are related to lower spatial frequency. Since human's eye is more sensitive to grosser details than finer details, DCT provides a mean to separate the less important information. Inside the DCT matrix, as we move from the upper left corner to the lower right corner, the values change from the grosser detailed information to finer details information. The most top left value corresponds to i=j=0 and is called the DC coefficient.

After the 64 DCT coefficients are generated they are quantized.

2.2.1.4 Quantization

Each DCT coefficient is divided to a relevant quantizer step. The quantizer step is generated by the multiplication of a base quantizer step size q_s (sometimes called MQUANT) and a weighting matrix shown below.

The matrix shows, as expected, a strong bias toward low frequencies, applying larger quantization size for higher frequencies than lower frequencies.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Figure 2-8: Intra-frame Quantizer Weighting Matrix

The result is then rounded to the nearest integer. The resulted integers are read in a zic-zac manner as shown below:

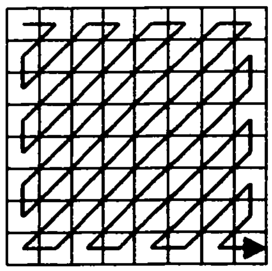


Figure 2-9 : Scanning Pattern

The reason for this method of reading is that after quantization more zeroes are likely to be appeared as we move from upper left to the right bottom of the matrix. Zig-zag reading enables these zeroes to be read consecutively, providing a long series of zeroes which could be effectively compressed using a combination of Huffman and Runlength coding. The DC coefficients are differentially encoded relative to the previous value. Since DC coefficients are large and since they are not changed too much relative to the DC values of their neighbors, this will lead to more compression.

The DC coefficients are differentially encoded relative to the previous value. Since DC coefficients are large and since they are not changed too much relative to the DC values of their neighbors, this will lead to more compression.

2.2.1.5 Predicted coding

Normally in a video sequence, there is a strong correlation between consecutive frames, so that a frame could be well predicted by applying movements to some regions of the previous frame. This idea is applied in predicted coding. In this mode, our coding unit is a Macro Block (MB), which is a region of 16*16 pixels (or 4 blocks). The method is that in the previous "anchor

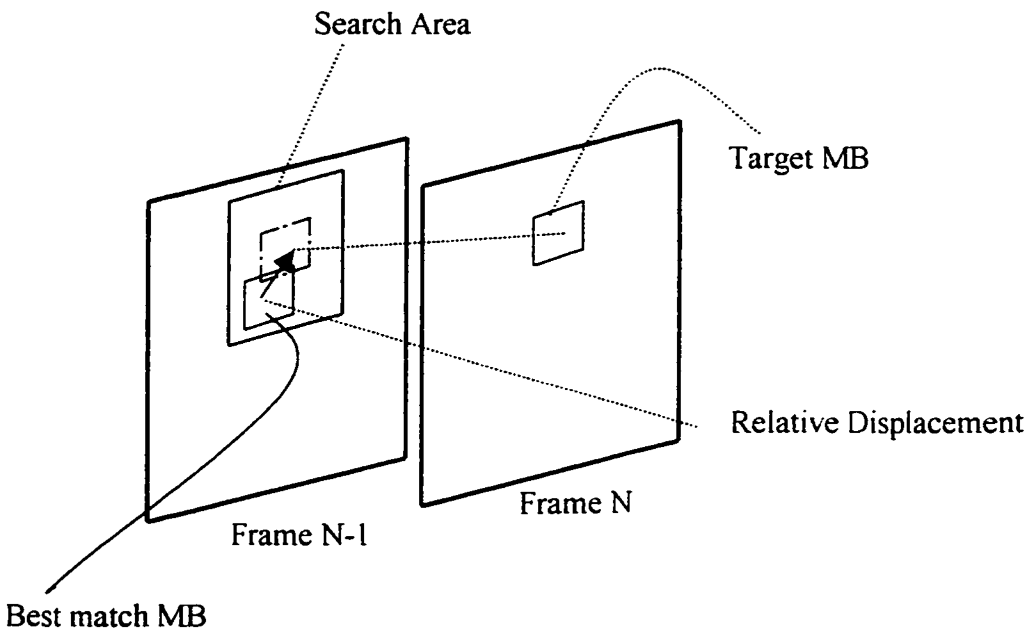


Figure 2-10: Applying Prediction to make new frames

frame" a region of 32*32 pixels is searched to find a suitable match for the MB. A common measure for matching that can easily be determined is some of absolute value of the pixel difference in the two blocks. If this match is found, Inter frame coding (motion compensation) is applied as below:

Motion vectors are applied to the best match MB, then the difference between this prediction and the original MB is calculated and *Intracoding* -as described above- is applied to this difference. The only difference is that for Intracoding of the error (difference between the MB that is to be encoded and the prediction), the quantization table, unlike the quantization table of I frame is uniform and all the values are the same (16 for example) for all the coefficients. Since the difference normally contains much less information than the target MB, this results in a considerable compression. The motion vectors are also variable bit coded and are sent together with the Intracoded of the difference.

In case that a suitable prediction is not found in the search region, the MB is coded completely as Intracoded.

2.2.1.6 Bi-directional Prediction

For B frames, prediction is applied to the previous and next anchor frames and the difference is calculated between the target MB and the average of the bi-directional predictions. This will result in even better compression for the difference between MB's.

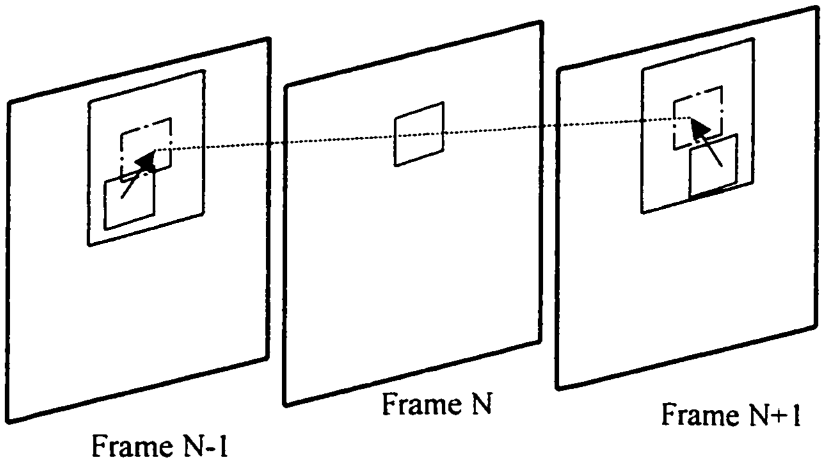


Figure 2-11: Bi-directional Frames

It is possible that a MB in a B frame is encoded from just one anchor frame or to be encoded intra-coding.

2.2.2 MPEG video structure

MPEG video has a complex hierarchy structure. A sequence of video comprises of several GoP (Group of Pictures). Each GoP composes of a sequence of pictures or frames in a specified order. Each picture (or frame) consists of several horizontal slices. Each slice consists of several Macro Blocks (MB) and each MB contains 6 blocks, two for chrominance information and four blocks for luminance information. The fact that human eye can differentiate better between the change in the brightness than change in the color, makes it possible to down sample the color information, so that every 4 pixels of luminance share the same chrominance information (which is the average of chrominance information of the 4 pixels). Hence each MB includes one 16 by 16 pixels (or four 8 by 8 pixels) of luminance information plus two 8 by 8 pixels of chrominance information. While in MPEG-1 a slice can begin and end anywhere in a frame (hence a frame can only have one slice), in MPEG-2, macroblock rows must start and end with at least one slice.

Each Sequence, GoP, Picture, Slice and Macro Block have their own non encoded headers which loss of it results in loss of appropriate data.

Sequence header, which precedes each GoP, among the other things specifies the size of each picture (frame) in the GoP (both in pixels and MB's), the two quantization matrices and the time space between the pictures. As we note, quantization matrices and frame rate could be changed for each GoP. This makes it possible to adapt the video stream over time.

GoP header is located at the beginning of each GoP and specifies number of frames for the GoP and synchronization information (which says that when the GoP should be played relative to the beginning of the sequence).

Each picture is given by a picture header followed by a set of slices. The picture header specifies the type of the frame (I, P or B) as well as the picture specific quantization table.

Slice header specifies the location of the slice within the frame. It also again enables the change of quantization table, this time by change of the base quantizer factor q_z.

MB header specifies the address of the MB within the frame, plus a quantizer step size for that macroblock which can be changed when it is preferred that a different quantizer step size than the corresponding slice to be used..

2.2.3 Dependency of the frames in MPEG GoP

As stated above, I frames are encoded individually and independently, which makes them suitable for randomly accessed points in a sequence of video. Also I frame are resynchronization points if a GoP is destroyed for any reason. I frames are typically largest size frames among MPEG frames.

P frames on the other hand are made through applying motion vectors to macroblocks of the previous I or P frame, hence their existence is dependant to existence of their previous I or P frames and loss of the previous I or P frame will cause loss of independent P frame. Although some macroblocks in a P frame may have been made by intra-coding, but generally the P frame is mostly made of predictively in order to allow more compression.

B frames are made based on bi-directional predictive coding of the previous and next anchor frames and are completely dependant on these frames, so loss of an anchor frame will cause loss of the dependant B frames too.

Introducing B frames have some advantages among them we can name better compression as described above and noise reduction. Noise reduction is achieved due to the fact that when there is a good prediction in both preceding and subsequent anchor pictures, then averaging the two predictors reduces noise. On the other hand B frames increases motion estimation complexity, not only because two anchor frames should be searched, but also since the space between anchor frames is increased by introducing B frames, then the search area should be increased accordingly. The reason is that the predicted Macro Block moves more as the temporal space increases. Furthermore, since for decoding a B frame, its next anchor frame should have been received by the decoder, not only this introduces delay at the decoder side but also it makes necessary buffering of previous anchor frame.

2.2.4 MPEG-1 and MPEG-2

In MPEG-2 (unlike MPEG-1) it is not necessary that a GoP structure to be followed precisely. Which means that in MPEG-2 we may insert an I frame or P frame whenever deems necessary. MPEG-1 generates long packets of bit stream to minimize header overhead, hence making it suitable for software based encoders. Large packets of MPEG-1 video stream on the other hand makes it more susceptible to errors, hence limiting its application to error free mediums as computer multimedia applications. MPEG-2 has the option of the large packets of MPEG-1 or shorter packets more suitable for error prone environments like communication networks. MPEG-2 also is able to provide support for larger screen formats as well as supporting field sub sampling of interlaced video making it more suitable for TV applications.

While the basic approaches of video compression of MPEG-1 (chrominance sub-sampling, DCT processing, motion estimation) have been used by MPEG-2 too, some enhancements have been included to provide support for new multimedia technology. For example:

-MPEG-2 supports alternative vertical zig-zag pattern for scanning DCT coefficients.

-MPEG-2 provides special concealment vectors used for error concealment of intra-coded MB's (more will be described in the next chapters).

-Accuracy of motion vectors has been added to half a pixel in MPEG-2.

Enhancements made to MPEG-2 have made it suitable for high definition television and the digital broadcasting of multiple television channels.

2.2.5 Variable Bit Rate (VBR) and constant Bit Rate (CBR) encoding

MPEG compression techniques could be used to achieve different parameters in terms of not only compression ratio, but also how the encoder delivers the bit stream. In other words, it is possible to force the encoder to deliver a constant bit rate output or variable bit rate (but with constant quality) stream. Constant Bit Rate (CBR) streaming means continues transfer of coded data with a constant rate, which is regulated through use of a feedback from an output buffer to the encoder. The CBR bit stream is delivered to the channel through this buffer. In CBR quality of frames is not the same, but the average bit rate is tried to be kept constant by changing the quantization level. The challenge here is to try not the buffer to overflow and change of the quantization step causes the change in the quality of pictures.

In VBR, unlike CBR, the quantization is kept constant hence providing the same quality for each picture. But since frames have different sizes and the size of a frame not only depends on the method of compression (Intra-frame, predictive), but also changes with the amount of activities in a scene, the out put stream can significantly change in bit rate. Since here the encoder should not change its quantization step to prevent overflow, OL-VBR also provides less encoding time in the expense of more difficult stream management as the rate is completely unpredictable. This means that in the OL-VBR, we deal with a more bursty output stream.

2.2.6 MPEG-2 Systems Layer

The MPEG-2 standard is divided into two main layers:

- Compression layer
- Systems layer

The Compression layer handles compression of the audio and video streams. The processing of this layer generates the so-called elementary streams (ES). This is the output of the video and audio encoders.

The Systems layer in MPEG-2 is responsible for combining one or more elementary streams of video and audio as well as other data into one single stream or multiple streams, which are suitable for storage or transmission. The Systems layer supports five basic functions, as described in MPEG-2 System:

- Synchronization of multiple compressed streams on decoding,
- Interleaving of multiple compressed streams into a single stream,
- Initialization of buffering for decoding start up,
- Continuous buffer management,
- Time identification.

A model of the Systems layer on the encoding side is shown in Figure 2-12. Each elementary stream, generated by the video and audio encoders, are first mapped into so-called Packetized Elementary Stream (PES) packets . The headers in the PES packets hold among other things timing information when to decode and display the elementary stream. Another rather important functionality is the possibility to indicate the data rate of the stream, which is used to determine the rate at which the stream should enter the decoding system

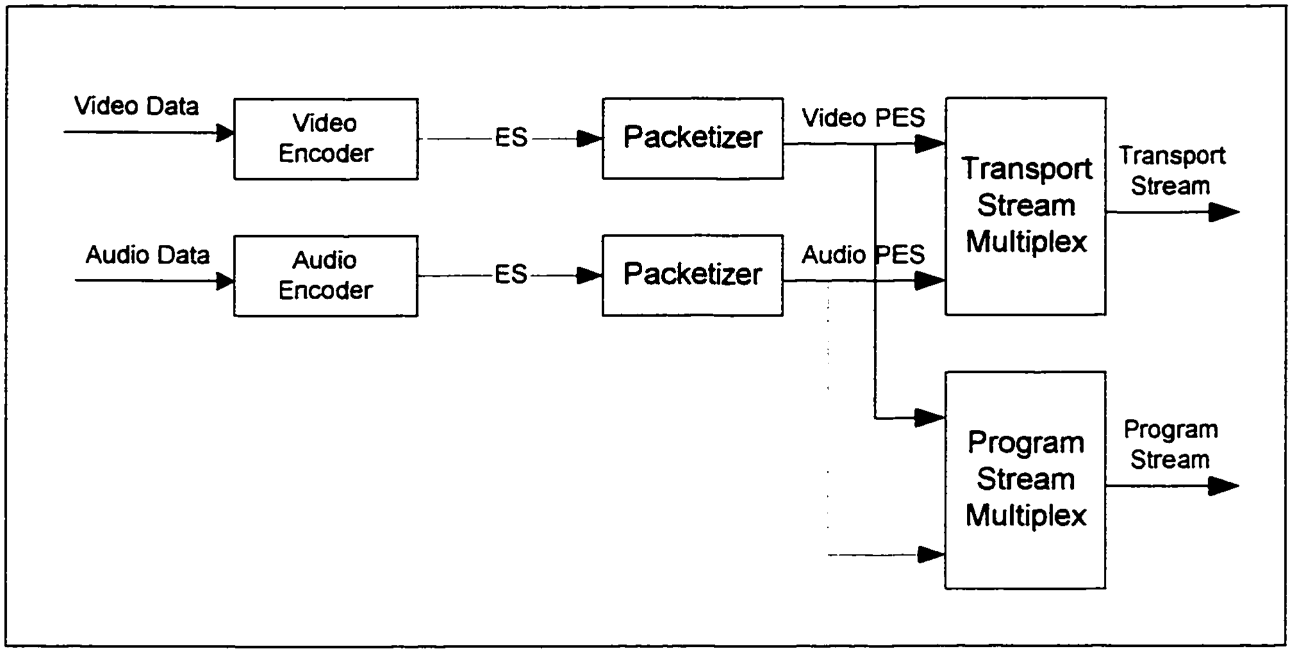


Figure 2-12: Model for MPEG-2 Systems in an implementation, where either of the Transport stream or the Program stream is used.

The Packetized elementary streams (PES) are multiplexed into either a program stream (PS) or a transport stream (TS), (Figure 2-14). A program stream supports only one program, whereas a transport stream may include multiple programs. Elementary streams of a single program typically share a common time base. The time base is the clock that determines among other things the sampling instances of the audio and video signals and is used when the elementary streams are generated. A program can for example be a television channel including a video stream and an associated audio stream.

In program streams, only elementary streams with common time base are multiplexed. Program streams are designed for use in almost error-free environments and are suitable for applications, which may involve software processing. Program stream packets may be of variable and relatively great length.

Both elementary streams with common time base (programs) and elementary streams with independent time base can be multiplexed into transport streams.

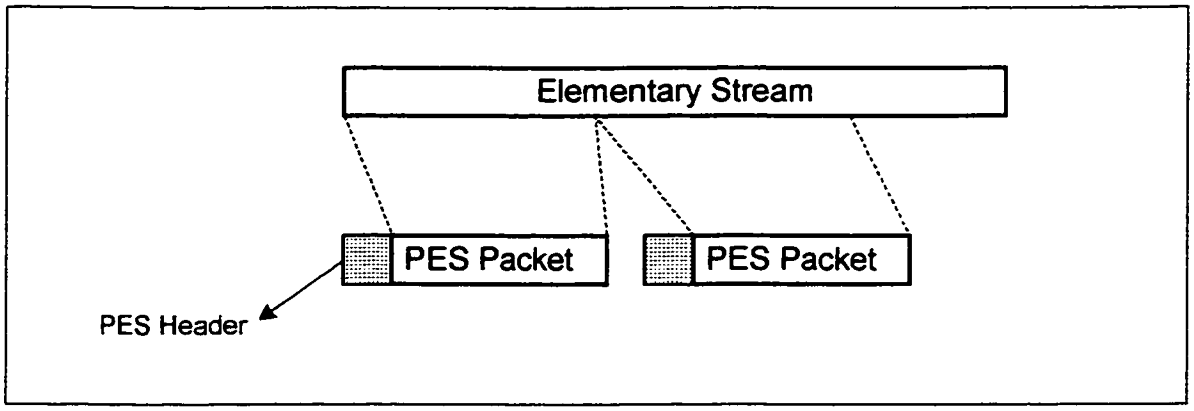


Figure 2-13: Putting ES in PES

Transport streams are designed for use in environments where errors are probable, such as storage or transmission in lossy or noisy media. Transport stream packets are fixed Size, 188 bytes long.

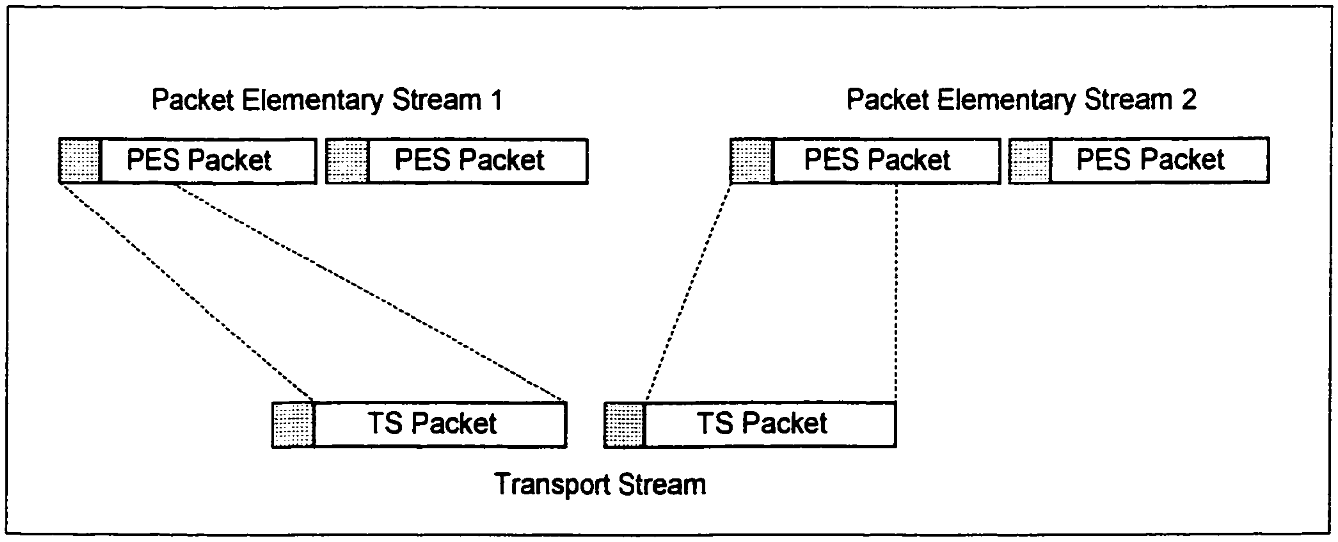


Figure 2-14: Mapping of two PES packets into one TS packet

3 Issues on transmitting of MPEG video over IP networks

Transmission of MPEG video over IP networks brings up some interesting issues for research such as different adaptation policies, effect of packet loss on the video stream, optimum packet size, etc.

In this chapter we briefly address some of the topics, which are necessary for understanding the remaining part of this thesis. We also briefly glance at the works that have been done on them so far.

3.1 Adaptive Streaming and Filtering

When MPEG video streams are put in IP packets and are transmitted over a best effort network, like any other packets, they are subject to loss/delay arising from nature of these networks. However, due to the hierarchical nature of MPEG video, effect of lost packets on the received video can be different. Exactly how a packet loss will affect the video clip depends on its location in MPEG hierarchy and will be addressed in section 3.2.

To keep the Packet Loss Ratio (PLR) of the streaming video below a certain level one way is to adapt the MPEG stream's required bandwidth to the changing condition of the network. The idea is that since we are dealing with a network that its bandwidth is shared among many users, there is no guarantee that this network can fulfill the requirements of all users. So to avoid congestion and unacceptable delays (which are not tolerated by the real time applications beyond a certain level), an adaptive service continually monitors the condition of network and controls the bit arrival rate to the network. This means that based on an agreed policy among sender(s) and receiver(s), it is preferred that instead of reception of an uncontrolled/degraded video due to lack of shared sources (or heterogeneity of senders and/or receivers as will be described later), a "controlled degradation" to be applied to maintain the QoS at a certain level.

It is not only the network's available bandwidth that can cause degradation, but also sender/receiver sides can be the sources of degradation (Figure 3-1). In other words, in a live streaming of video over Best Effort network, the three main bottlenecks are sender/receiver/network as described below [15].

3.1.1 Server generated degradation

The sender side (server) may not be fast enough to be able to send the stream as it is delivered to it by the encoder ("encoder rate"). It may also have changing playing capability because of its dispute between I/O resources and CPU. This source of degradation is independent of network/receiver conditions, and even if both network and receivers cause no degradation, the receiver(s) may experience it. With the powerful servers available today, this source of degradation is less important and we will disregard it in the rest of this study.

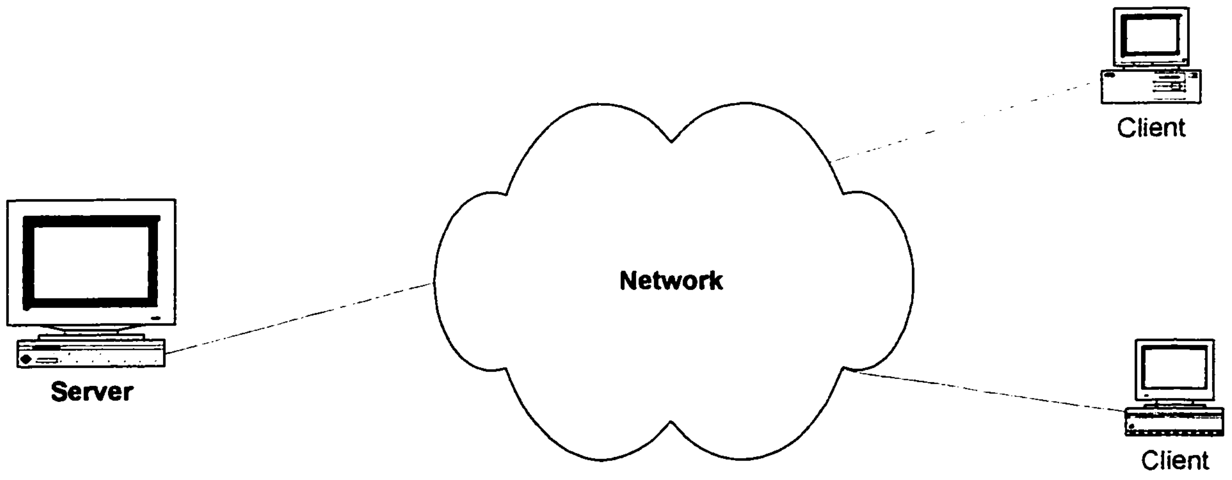


Figure 3-1: The 3 causes of degradation in a BE network

3.1.2 Network generated degradation

The network is subject to receiving undetermined (random) traffic from the users other than MPEG users; so it has fluctuating background traffic ("background" from the "MPEG user" point of view).

The background traffic causes different queuing delays for MPEG traffic, hence causing delay jitter and necessitating the use of a dejitter buffer. The network can also introduce bursty background traffic, which can cause the packets to be discarded by the routers due to lack of capacity of their buffers. The background traffic can also cause a packet to reach the destination with such a long delay that it is not useful anymore; hence it is considered as a lost packet. This will be described in more detail later in 5.2.4.

This source of degradation is the most challenging cause of degradation to cope with in a BE scenario, but could be solved more easily in a network with reservation capability mechanisms.

3.1.3 Receiver generated degradation

The receiver may be slow so that it is not able to decode and play the stream as it is received. In this case it is probable that the receiver side's capability to capture and play the stream changes with time if it should compete for the shared resources. This may make the situation more difficult when there are many receivers in a multicast scenario and some of them are "slow". The problem obviously cannot be addressed by adaptation at the sender side, e.g., reduction of bit arrival rate at the sender side, as in this case "powerful" receivers are forced to slow-down themselves with the slowest receiver. This brings the issue of "fairness" in a multicast environment as discussed in [16].

3.1.4 Adaptation techniques

Adaptation is defined as the techniques that are used in the BE networks to deal with the variation of resources and to enhance the QoS of the received video. Adaptation techniques could be either source-driven or receiver-driven. [17]

3.1.4.1 Receiver Driven Adaptation

These techniques are used in multicast scenarios when many receivers want to receive a video stream from the same server. In a receiver-driven adaptation, the source sends different levels of bit rate corresponding to different qualities of the pictures called "layered streams". The receivers then receive a suitable level of encoded video depending on their available bandwidth.

To reduce computational time in the sender side and providing real-time adaptation, it is preferable that a process first decomposes the standard MPEG stream into a layered MPEG stream including one basic and some enhanced layers.

Receiver-driven Layered Multicast (RLM) is a well-known technique among the receiver driven adaptation techniques [18]. In this method the video stream is encoded so that it consists of one basic layer and one or several enhancement layers. The basic layer includes basic information about the clip, while adding enhancement layers adds the video details to it so that a basic layer together with all the enhancement layers provide a quality equal to the quality of the encoded video clip. The problem of heterogeneity among the receivers could be solved by this method so that the receivers subscribe to the layers based on their capabilities. Those that have less processing power subscribe only to the basic layer and so on. One may think of the basic layer as the encoded information in I frames and the enhancement layers as the information encoded by P and B frames.

3.1.4.2 Source Driven Adaptation

In a source-driven adaptation technique, the source somehow receives feedback from the receiver and/or the network and will adapt its rate to the requirements of them. Several source driven adaptation techniques have been proposed in the papers. Decreasing the bit arrival rate is achieved by change of one or several parameters. Ramanujan et al [17] divide the source-driven adaptation techniques to two groups of SNR adaptation and displayed frame rate adaptation. SNR adaptation is applying variation in bit arrival rate by change of:

- Range of DCT coefficients used through MPEG encoding. This technique is named "Spectral filtering".
- The precision by which the DCT coefficients are encoded, i.e., discarding LSBs of the quantized DCT coefficients. This technique is called "quantization filtering".

Some other techniques are:

- Color elimination of a color video clip.

- Dropping some frames out of each GoP of the MPEG video clip as described in [19]

The devices that perform such adaptation techniques are referred as "filters". Filters are defied as performing three functions: Selective, Transforming and Mixing. A selective filter decides whether to forward or drop certain packets based on some criteria. Essentially, the filter either drops sub-streams of hierarchically encoded streams before they reach particular client or is used to reduce frame rates. Transforming filters perform computations on a stream such as reducing color information from 24bits to 8 bits for example. Filters can be placed in different places. While some are placed within the networks, some are used by clients or severs depending the adaptation policy. Mixers mix multiple streams into a new stream, which can involve complex processing of the stream. [20]

This research concentrates on the source driven adaptation techniques and hereinafter, adaptation means source-driven adaptation, unless otherwise stated.

3.2 Error Propagation in MPEG video stream

MPEG video streams, as described previously, have a complex hierarchical structure. Generally speaking, MPEG video streams carry two kinds of information: Syntactic and Semantic.

Syntactic information includes such data as headers and system information. Semantic information on the other hand includes pure video information (motion vectors, DCT coefficients, etc.). Any error, which happens during transmission of these two kinds of information, has different impacts on the received video. Furthermore, degradation in quality of received video also depends on the location of the lost (or in error) semantic data as MPEG uses predictive and bi-directional encoding techniques for compression.

One study [21] shows a random loss of 1% of IP packets carrying MPEG video information through Internet can be translated into as high as 10% of damaged video frames. Others [22] noticed that a 3% loss of IP packets is propagated to damage 30% of frames.

3.2.1 Syntactic Error Propagation

Decoding information carrying headers are added to the beginning of each Sequence, GoP, Frame and Slice. Loss of any of these headers normally causes that the corresponding element of information to be useless.

Figure 3-2 shows different cases of syntactic error propagation in a sequence of MPEG video. This kind of error is generally more difficult to be recovered.

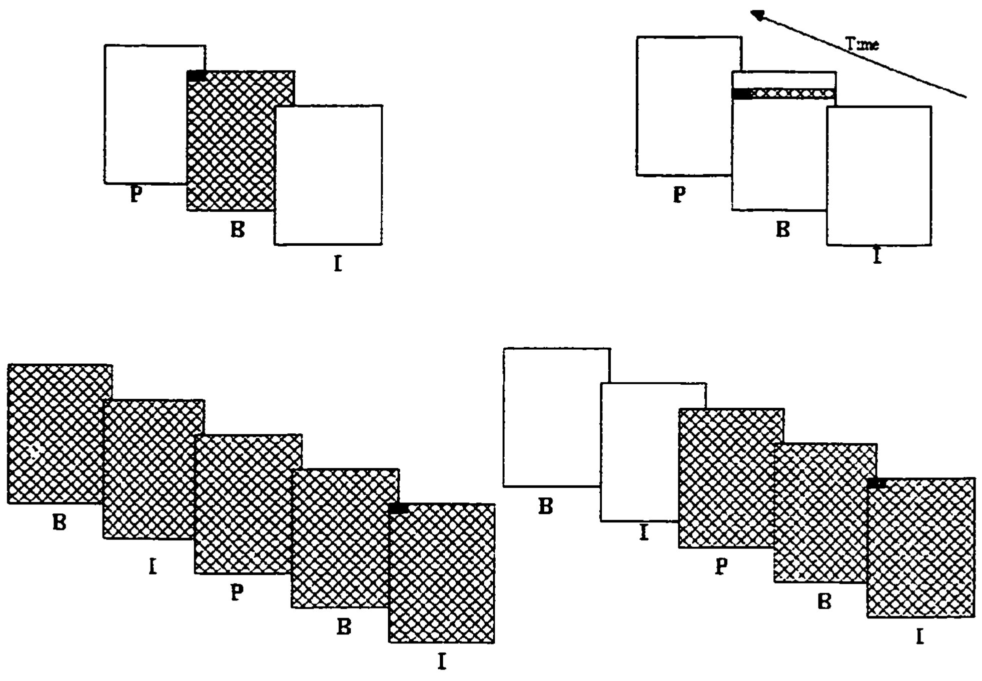


Figure 3-2: Syntactic error propagation in a sequence of MPEG video

3.2.2 Semantic error propagation

Semantic error propagation are resulted from loss or error in pure video information.

We can divide the propagation of errors in MPEG video to two groups: temporal error propagation and spatial error propagation. [23]

Temporal error propagation results when a macroblock belonging to an anchor frame is received in error or is lost completely. In this case, the inter-coded frames that applied motion vectors to this macroblock for prediction of their own macroblock will not able to decode that part of frame anymore

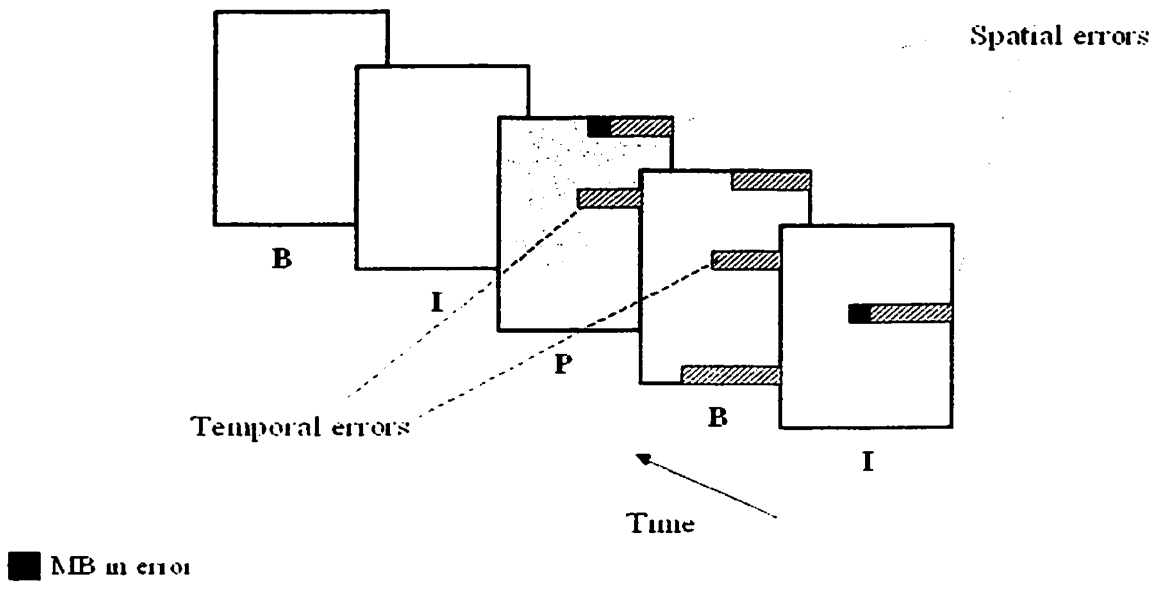


Figure 3-3: Semantic error propagation in a sequence of MPEG video

Spatial error propagation is the result of spreading an error in a frame to the next Synchronization point. This happens mainly due to the use of differential coding and Run Length Encoding(RLE). Spatial errors can happen within any kind of frame, however only errors in anchor frames lead to temporal errors in the inter-coded frames. I frames' headers are resynchronization points for temporal error propagation and slices' headers are resynchronization points for spatial error propagation.

3.2.3 Error Concealment Techniques

In order to maintain a certain amount of acceptable quality for the user, means should be provided to conceal the errors happened during the transmission. These error

concealment techniques include, for example, spatial interpolation and temporal interpolation.
The MPEG-2 standard proposes an elementary error concealment algorithm based on motion compensation. Basically, it estimates the motion vectors for the lost macroblock by using the motion vectors of neighboring macroblocks in the affected picture (provided that these have not been lost). This improves the concealment in moving picture area. However, there is an obvious problem with errors in macroblocks whose neighboring macroblocks are intra-coded, because there are ordinarily no motion vectors associated with them.
In general, error concealment techniques may effectively decrease the sensitivity to the data loss. However, none of these techniques is perfect. Data loss may still involve annoying degradation in the decoded video.

3.3 Packetization of MPEG streams in RTP payloads

IETF RFC 2250 describes packetization of MPEG video for transmission over IP networks. It recommends a packetization scheme for MPEG video and audio stream using RTP. The standard not only describes the encapsulation of "pure" video or audio information ("Elementary Streams" or ES), but also proposes a way of encapsulating higher layer MPEG (i.e., Program Stream, Transport Stream) in RTP. [24]
As described previously, Elementary Streams are separately put into packets at the encoder. These packets are called "Packetized Elementary Streams" (PES) and among many information that they carry in their headers, they also carry "time stamps" named as "Presentation Time Stamp" (PTS) and Decoding Time Stamp (DTS). The PTS is needed to ensure that the audio and video signals are provided to the loudspeaker and the television screen respectively at the same time. The DTS indicates when the received data has to be decoded. Note that due to inter coding nature of B and P frames, DTS for these frames is done when their respective anchor frames have already been decoded, while the time they are presented to the viewer (PTS) should follow the order the frames make a GoP.

When dealing with Elementary Streams, a distinct RTP payload is assigned to MPEG1/MPEG2 video and MPEG1/MPEG2 audio respectively. No information about the version of MPEG is necessary, as this information already exists in PES streams.

Since MPEG video pictures normally have large sizes, they are fragmented into smaller packets equal to MTU of the LAN/WAN they are traveling. Based on RFC 2250 recommendation, in fragmentation, the sequence header, GoP header and picture header should always be at the beginning of RTP payload. Also each ES should be completely contained within the packet.

RFC 2250 recommends that an integral number of slices to be put inside a RTP packet, however, It does not specify any further detail or obligation about the packetization of RTP payload.

Two approaches are described in the standard. The first one specifies how to packetsize MPEG-2 Program streams (PS), Transport streams (TS) and MPEG-1 system streams. The second gives a specification on how to encapsulate MPEG-1/MPEG-2 Elementary streams (ES) directly into RTP packets. The former method then relies on the MPEG systems layer for multiplexing, whereas the latter method makes use of multiplexing at the UDP and IP layers.

3.3.1 RTP Encapsulation of MPEG-2 Transport Stream

Each TS packet is directly mapped into the RTP payload. To maximize the utilization, multiple TS packets are aggregated into a single RTP packet. The RTP payload will contain an integral number of TS packets. In the Ethernet case, where the MTU is 1500 bytes, there will be seven TS packets in each RTP payload (RTP payload size=1316), and every IP packet will have a size of 1384 bytes.

In the MPEG-2 Program Stream case there is no packetization restrictions. The PS is treated as a packetised stream of bytes. In Figure 3-4, the protocol architecture for TS over IP networks is illustrated. For each protocol, it is also shown, which TCP/IP protocols layer it belongs to. In the TCP/IP suite the MPEG-2 Systems layer is considered to belong to the Application layer.

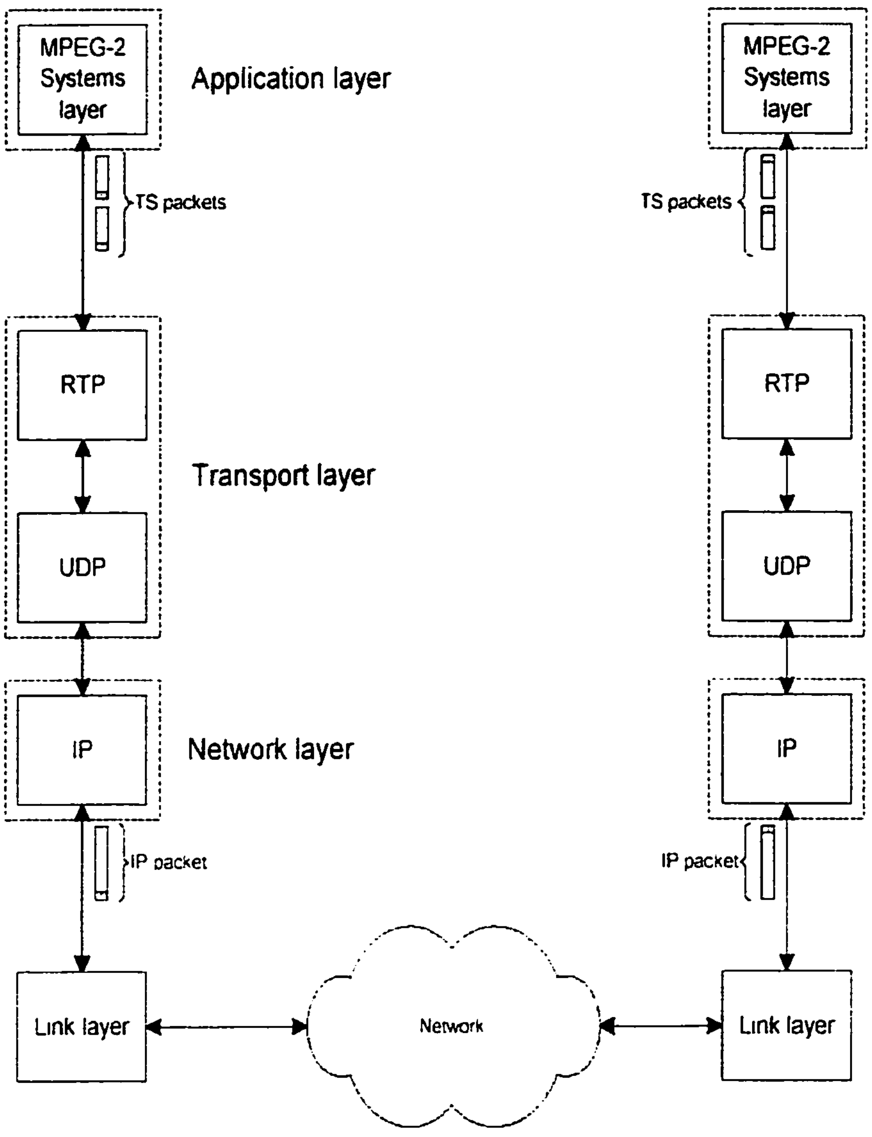


Figure 3-4: Mapping of TS packets into RTP payload.

3.3.2 RTP Encapsulation of MPEG-2 Elementary Stream

The second approach described in RFC2250 is to pocketsize MPEG-1/MPEG-2 elementary streams (ES) directly into RTP packets. Audio ES and Video ES are sent in different streams and different payload type is assigned to them. Both audio and video streams have their own payload header that provides the information that the MPEG-2 System layer normally provides. In this case the timestamp in the RTP header represents the presentation timestamps (PTS) in MPEG-2 Systems layer. This timestamp i used both for reduction of jitter and in the decoding process.

In the next chapters we will investigate effects of RTPs' packets size in a best effort network that uses a source driven adaptation technique for improving video QoS.

3.4 Network Performance Measures

Whether we are dealing with multicast (multi-peer communication) or unicast of MPEG video over IP, the quality of service (QoS) should have an acceptable level.

There are two types of criteria that can be used for the evaluation of the video quality: subjective and objective [25],[26]. Subjective video quality assessment is the most reliable video quality measurement method. A group of viewers is selected and gathered in a room with precisely specified environment. Source video and the processed video are presented in pairs to the viewers who are expected to grade the video quality. The subjective measures use rating scales such as goodness scale and impairment scales. Although subjective evaluations are most reliable measures because the user is a human observer, they are costly and time consuming. Furthermore, subjective video quality measurement cannot provide real-time quality monitoring for real-time video applications

Objective video quality measurements, although not as accurate, can be conducted in the background without conflicting with the end user. However, they do not match well to the characteristics of human visual system. Objective measures are performed using mathematical equation. Mean squared error (MSE), peak signal to noise ratio (PSNR), Packet Loss Ratio (PLR) are common objective measures

Glitch is another tool for performance measure of the network. Glitch is defined as an event that is said to begin when a portion of a frame is not displayed due to unavailability of data, while its preceding frame is fully displayed, and duration of a glitch is the time (or number of frames) between two consecutive fully displayed frames. Glitch spatial extent is the percentage of undisplayed portions in the frame. How the glitch and its duration affect the perceived quality depends on the content of the video. In fact some glitches may even can not be perceived by the viewer or affect the perceived quality very little specially when the video clip presents a less changing scene like a video conferencing. In [26] it is shown that the glitch rate is mainly affected by the network type, traffic load, encoder control scheme and video content while glitch spatial extent and duration are mainly affected by the network type and the video encoding scheme (VBR/CBR).

4 A source driven adaptation technique

In this chapter we will describe a source driven adaptation mechanism for streaming of MPEG video over BE IP networks. Meanwhile, we propose two different packetization schemes consistent with RFC2250 for putting video MPEG information inside RTP packets and will investigate how these different schemes will affect the PLR before and after adaptation. What is discussed in this chapter will be simulated in the next chapter to verify the results.

4.1 Investigating packetization schemes within the adaptation techniques

Since this research focuses on MPEG *video*, we consider the second method of encapsulating MPEG video recommended by RFC2205 as described in 3.3.2, i.e., we assume that video Elementary Streams (ES) are put into RTP packets and the header carries all necessary information including PTS information. We should note that the first packetization scheme recommended by RFC2250 (described in 3.3.1) i.e., packetization of MPEG Transport Stream (TS) in RTP packets, does not allow us to concentrate on MPEG video, as TS packets contain both audio and video ES as described in 2.2.6 and we will not be able to differentiate between video and audio information. Also it is worth mentioning that another standard (RFC2343) describes guidelines for packing audio *and* video *ES* together in the same RTP packets (known as bundled MPEG), which is useful if for some reason it is preferred to encapsulate video and audio ES together.

As described previously, in the source driven adaptation policies, the sender adapts its input rate with the condition of the network, and through the feedback that it receives from the receiver and/or network, adapts this input rate. The way that a source can change its bit rate was described previously in 3.1.4.2. We use this technique in this research and through our simulation.

4.1.1 Figure of merit

Among the objective performance measure described in 3.4, we select the Packet Loss Ratio (PLR) – a merit that could be used through a simulation too- as the mean for controlling input rate of the source. Based on this value, which is measured *periodically*, we adapt the input bit rate to control loss ratio.

Packet Loss Ratio (PLR) is defined as the ratio of the corrupted RTP/UDP/IP packets to the total packets that the sender sends into the network. A packet that is received corrupted (or is lost) may cause temporal/spatial error propagation as described in 3.2 and thus will cause the other packets that have reached the destination "untouched" to become useless. In this case (which due to interdependency of MPEG frames) is expected to occur frequently), total number of damaged packets is calculated in Packet Loss Ratio. Although this way we have considered not only the actual lost packets but also the in-error/lost packets too, we are still at network-level monitoring of QoS.

4.2 The adaptation technique

We use a combination of frame dropping (as described below) and change of the MPEG quantizer scale size as a mean to control input rate of the MPEG video information.

4.2.1 Frame dropping

Frame dropping is defined as using a filter after the encoder so that it filters some frames based on a policy. By responding to the increasing PLR at the receiver side, the filter drops some frames of the GoP.

A good frame dropping policy causes the least possible degradation in the video stream. This happens when we have less variation in time space between the frames. The higher the PLR, the more the number of the frames that are to be dropped.

To explain the frame dropping policy that we have proposed, assume that our Group of Picture (GoP) is composed of 15 frames with the order **IBBPBBPBBPBBPBB**, which are generated and sent during 500 ms. So our frame rate is 30 frames/sec and each frame is sent every 33.3 ms. (See Figure 4-1)

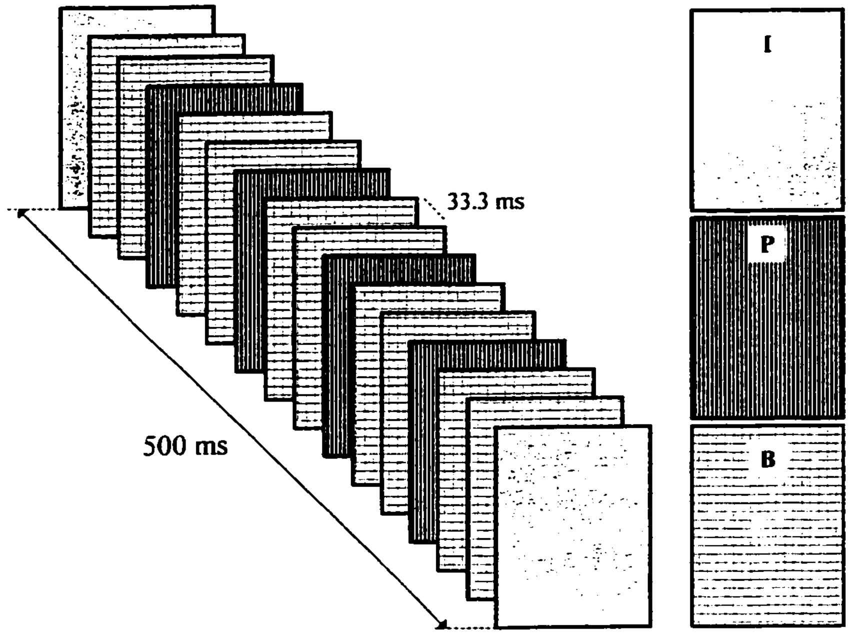


Figure 4-1: GoP structure

As described in chapter two I frames are the largest frames and B frames are the smallest size frames. Obviously I frames cannot be dropped as they are anchor frames that all other frames of the GoP are calculated based on them. Also they represent resynchronization points if a part of streamed video is damaged.

P frames are created by applying prediction to the I frame or previous P frame and they are anchor frames too, which help bi-directional prediction to be used for calculation of B frames, which are completely independent frames. This implies that the frame dropping should start with these independent frames, which does not affect other frames.

The first stage of frame dropping in our adaptation scheme starts with dropping the 2nd B frames in the GoP as depicted in Figure 4-2, i.e., five B frames are dropped from each GoP. So the new sequence that will be sent in the network will be in the form of **IBPBPBPBPB** sequence. At the receiver side, to improve perceived quality of video,

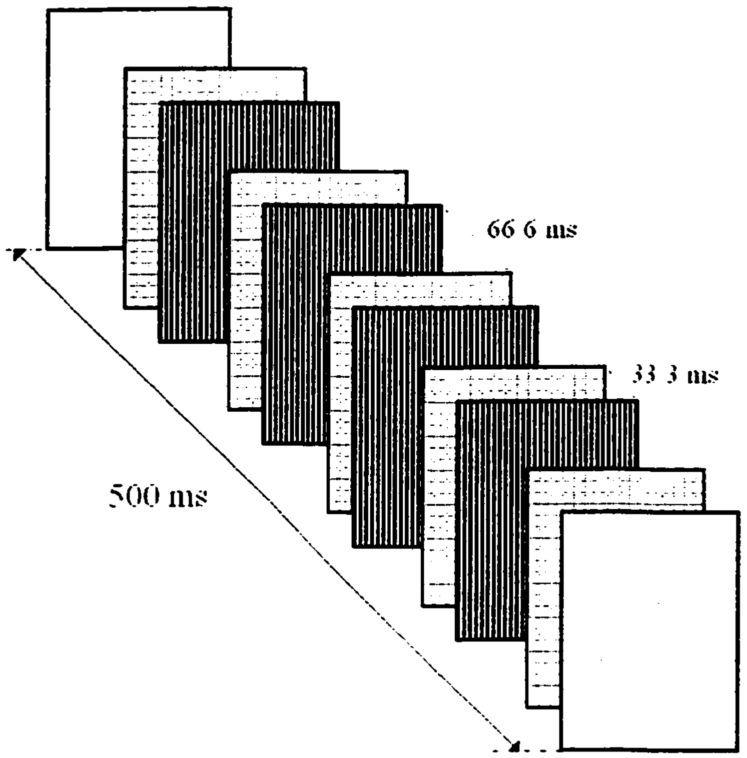


Figure 4-2: First stage of frame dropping

the previous frame could be repeated, or manipulation could be used to generate an estimation of the deleted frame. We do not address this subject as this is out of scope of this research.

The 2nd stage of frame dropping deletes all remaining B frames resulting in sequence of **IPPPP** and frame structure as shown in Figure 4-3. Note that at this stage the frames are located equally spaced in time as shown in the figure.

Although at this stage two third of the frames have been removed but since B frames are smaller in size comparing to I and P frames, *this does not translate to the same ratio in the bit arrival rate.* With a typical ratio of I/P/B frames as 7/3/1, at the 2nd stage only a reduction of around one third in the arrival rate has been achieved.

The next stage at frame dropping will be removing the last P frame, which does not have any affect on its previous frame resulting in a frame sequence of **IPPP** and the last stage

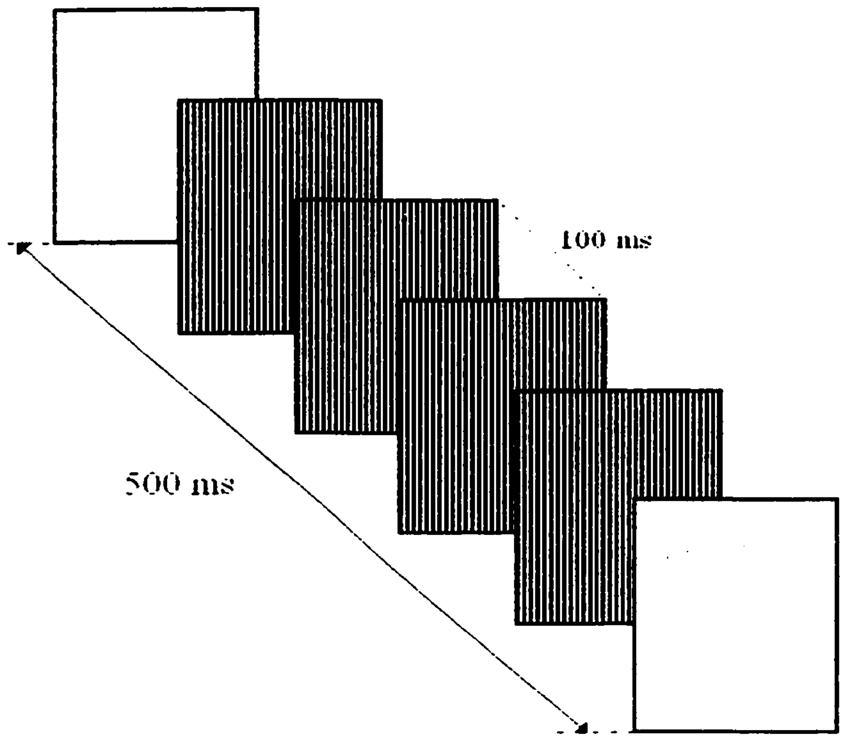


Figure 4-3: Second stage of frame dropping

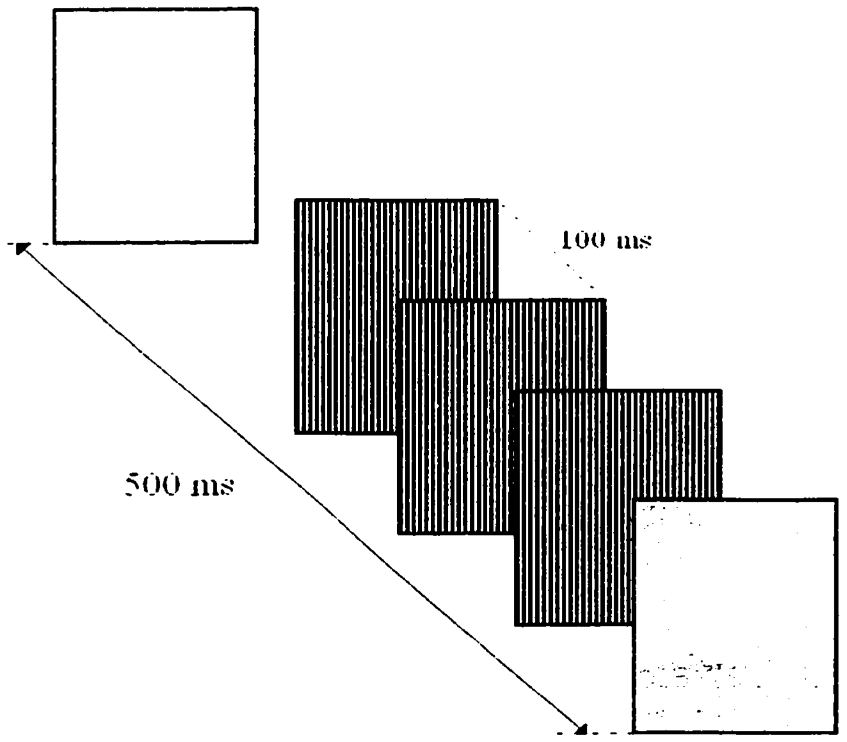


Figure 4-4: Third stage of frame dropping

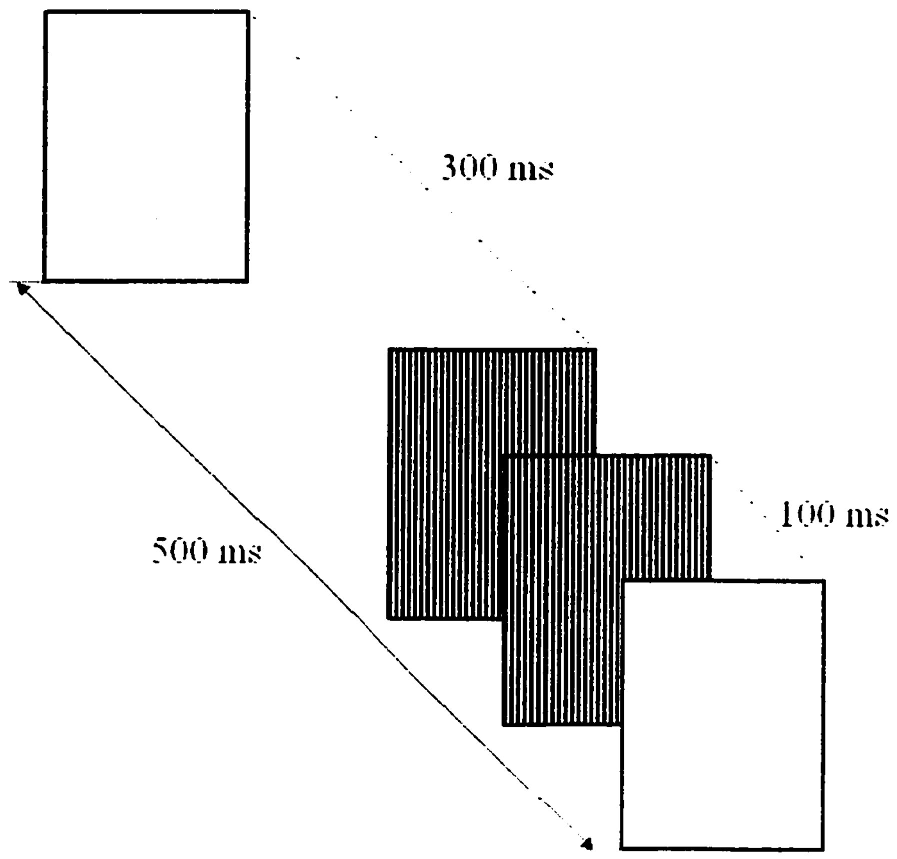


Figure 4-5: Fourth stage of frame dropping

is eliminating another P frame (obviously from the end of GoP). The proposed algorithm stops dropping frames at this stage and do not drop any frame anymore. So the minimum sequence of frames will be **IPP** and the minimum frame rate is 6 frames per second.

4.2.2 Changing quantizer step size

To be able to have more control on the sending bit rate, MPEG quantizer scaling factor described in chapter two is also changed as the second way of control on the MPEG video.

The multiplicative quantization scale factor, **quantizer_scale**, can be changed at the start of coding of each macroblock. The bit rate is very sensitive to the **quantizer_scale**, and variable quantization is an important tool for the picture quality and controlling bit rate. [13]

The combination of frame dropping and quantization level allow more granularity on the adaptation algorithm. How we have combined these two elements will be discussed in 5.2.10.

4.2.3 Packetization

As described in 4.1, we put the video Elementary Streams (ES) in the RTP packets. For the packetization we follow two different schemes.

- First we put a whole MPEG frame inside each RTP/UDP/IP packet before transmitting.
- In the second method we put one single slice inside each RTP/UDP/IP packet before transmitting it to the network.

Both of these packetizaion schemes are consistent with RFC 2250, as both satisfy the rule that there should be an integer number of slices in each packet. (Frames consist of seven slices in QSIF standard and 11 slices in SIF standard).

We will then investigate how these different packetization schemes will affect the PLR before and after applying the adaptation.

4.2.4 Effects of networks parameters

In our research, we also investigated the effects of network parameters on the PLR before and after applying adaptation for each of the proposed packetization schemes. These parameters are:

- Background traffic (or load) in the network (burstiness of traffic, packet size of traffic)
- Capacity of the links of the network
- Buffer size of the routers
- Dejitter buffer size of receivers

5 Simulation and results

In this chapter we describe the simulation that we used to investigate the source driven adaptation technique that we have chosen and investigate effectiveness of this technique and how the different packetization schemes affect video streaming before and after adaptation. We will describe traffic models, network configuration and topology, video traffic generation and other details of the simulations in this chapter. We will also provide some definitions that based on them, the results will be discussed later in the second part of this chapter.

5.1 Program

In this section we briefly describe the simulation program. Since the program is rather large (around 10000 lines) only some general information is provided.

The simulation program has been written in C++ using Microsoft Visual C++ version 6.0 under Windows NT environment. The outputs of the simulation are text files consisting of numeric values generated by the simulation program. We then use these values to generate the figures of the next chapters using MatLab version 5.3.

The simulation program starts with defining and building the structure of the simulation network. (The network structure will be described in the next section). This is done through definition of a class named **node** to define the properties of each router in the network. Each node has its relevant neighboring nodes. Each node consists of output buffers, which have certain capacity and are defined at this stage of the simulation and a decision-making fabric, which forwards the packets. The buffers are simulated through circular arrays [27] and two pointers keep track of **head** and **tail** of the queue. The output buffers are connected to the other routers through links with certain capacity and length.

The two most important functions of the **main()** program are:

generate() which generates MPEG video and data (background) traffic. Each call of this function generates MPEG traffic and data traffic for the next 500 ms of the simulation. (Although in real world as the traffic is generated it begins to travel, but it is impossible to simultaneously simulate generation and transfer, unless we have some knowledge about timing and size of the future traffic in advance.)

The corresponding time for data traffic (background traffic) that indicates exactly at what time the packets should be transmitted from the source are generated at this time. MPEG information are transmitted continuously but with variable bit rates.

transfer(): This function simulates "motion" or transfer of the packets through routers and links for one ms. So for each call of generate(), transfer() is called 500 times. The program looks at every packet each ms, i.e., at each ms a snapshot of all parts of the network (including packets at the network) are observed and the status of each packet is refreshed. This refreshing includes transfer of all the packets in buffers, links and routers one step forward. The movement of packets during each ms depends on capacity of the links, speed of routers, etc.

Functions of a links (transferring, applying link's rate capacity, applying propagation delay) has been simulated by introducing a (virtual) buffer (an array) inside each link which, gradually "sinks" the packets at the front end of the router's output buffers in each iteration of **transfer()** (based on the capacity of the link) and delivers it to the next router at appropriate time (based on the propagation delay of that link).

5.2 Simulation Assumptions

Figure 5-1 shows the network that we have used for the simulation and Figure 5-2 shows graph representation of it. As shown in Figure 5-2, eight "nodes" are connected to each other by several "links". To each node several "users" are connected which transmit and receive IP datagrams to and from each other in a unicast mode. In the case of "MPEG users", they represent an entity, which its functionality is to receive the video Elementary Streams from the decoder, pack them in RTP/UDP/IP packets, source fragment the packets as will be described later in 5.2.7, and deliver them to the first node to start their trip to their destination.

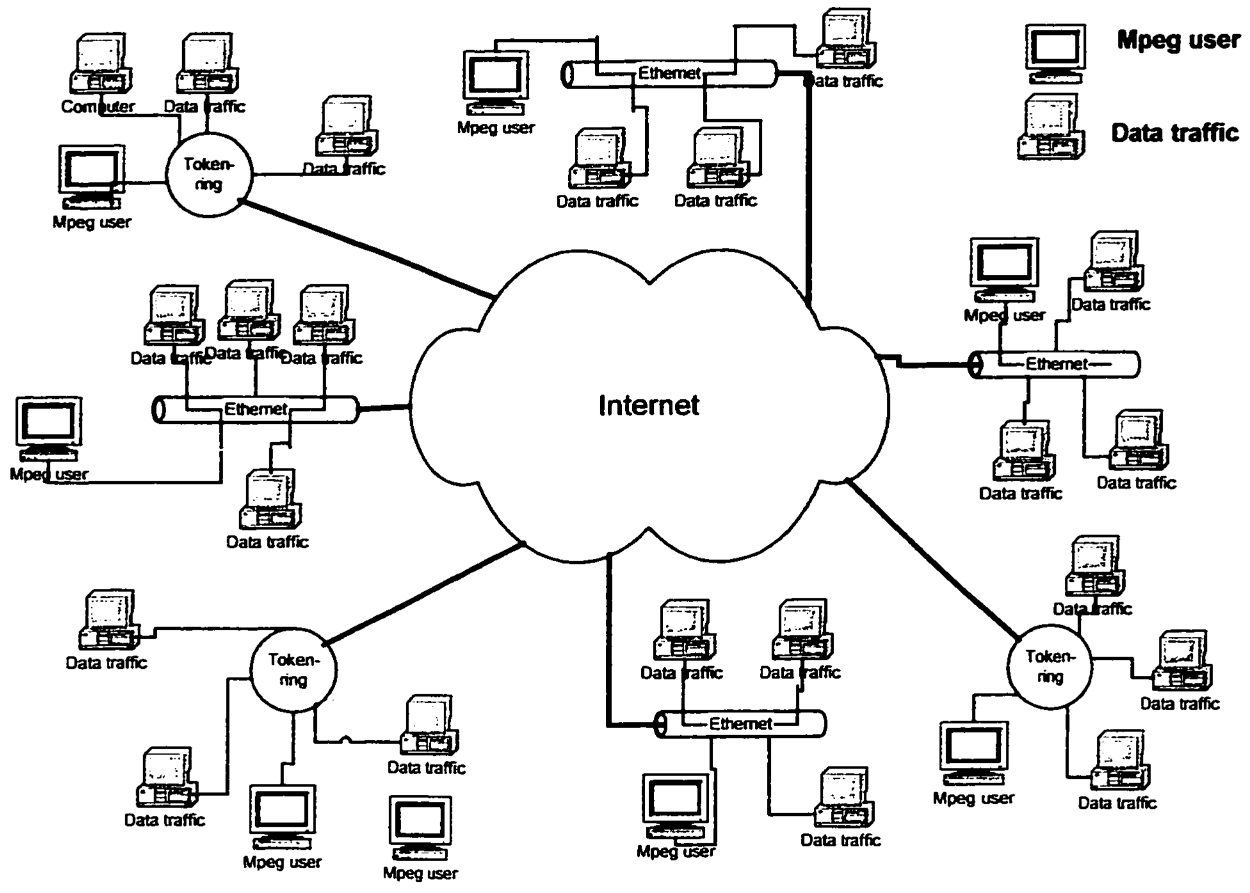


Figure 5-1: The network used for simulation

5.2.1 Traffic sources

5.2.1.1 Internet Background Traffic

To each node many "internet users" are connected. These users send data traffic to the nodes. Based on the destination of each packet, it is directed to the corresponding output buffers of that node. To simulate the aggregate traffic received by each buffer, one "traffic generator" for each buffer has been connected to each node. These generators generate data IP packets at exponentially distributed inter arrival times; hence their generation follows a Poisson's distribution. Poisson's process is considered to be a good model for the aggregate traffic of large number of similar and independent users like Internet users [28]. Each one of these 26 traffic generators actually represents the whole internet users that transmit to each buffer of each node. (see Figure 5-3).

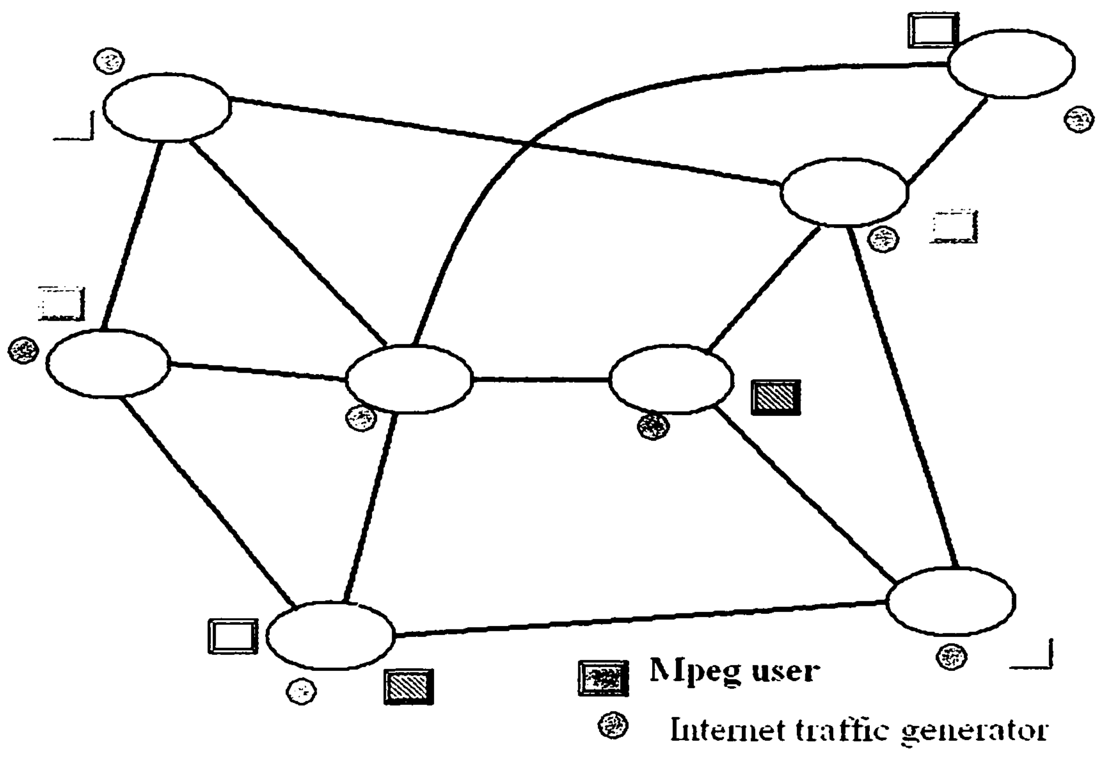


Figure 5-2: Graph representation of the network used for the simulation

Since it is believed that the aggregate Internet traffic that the users inject to the links follows a Poisson distribution [29], we have used one generator for each buffer of each node to simulate this aggregate traffic. We call the traffic that is generated by these 26 generators "Background traffic".

5.2.1.1.1 Load of the network

The rate at which the traffic is generated and the size of each IP packet could be changed in each run of simulation, but remain unchanged during one run. So the background traffic packet's size and arrival rate remain unchanged during one run of the simulation, but could be (and are changed) from run to run to investigate their effects on the results. We call these two parameters as "traffic rate" and "traffic packet's size".

How frequently the packets are generated, i.e., "traffic rate" simulates load of the network. In real world this parameter changes with the time of the day having the least value at around 4 AM.

The background packets are generated instantly as an "explosion" of bits and are put in the output buffer of the router to which they are directly connected. This means that, a packet is generated at an instance. With the power of today's PC's and considering the size of these packets in the simulation, this assumption is valid. This is an important attribute of the background traffic.

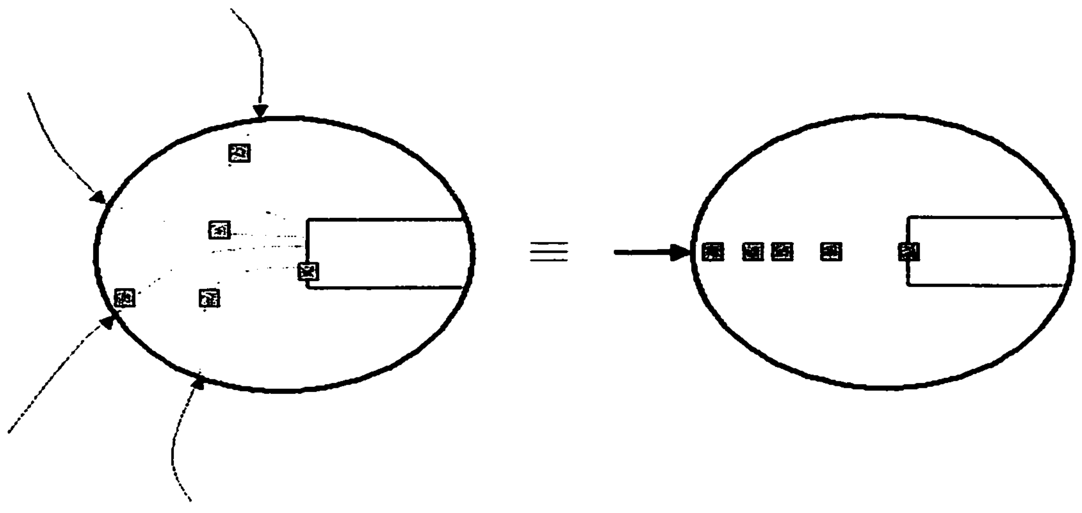


Figure 5-3: Simulation of aggregated Internet traffic at each link

5.2.1.2 MPEG Video Traffic

The eight video MPEG encoder generate a stream of video according to the following 15 frames Group of Picture (GoP) structure:

IBBPBBPBBPBBPBB

Each GoP corresponds to 500 ms, so our frame rate is 30 frames/Sec and each of the above frames are generated during 33.3 ms. We have assumed that the frames generated are according to QSIF standard. This assumption will have an effect on number of slices per frame as will be described later.

Each GoP corresponds to 500 ms, so our frame rate is 30 frames/Sec and each of the above frames are generated during 33.3 ms. We have assumed that the frames generated are according to QSIF standard. This assumption will have an effect on number of slices per frame as will be described later.
Our MPEG encoder has Variable Bit Rate (VBR) output, i.e., to keep the quality of all output frames constant, the MPEG quantizer step factor (MQUANT) is held constant for all frames. As described previously, although the frames are generated at equal intervals, they have different sizes. The size of the frames is not only a factor of the complexity of the scene they represent, but also depends on the inter-frame or intra-frame coding of them. I frames have the biggest size and B frames the smallest size.
In our simulation the sizes of the frames are randomly selected from a uniform random generator as follows:
I frames are between $\delta \times 3.75$ to $\delta \times 5.0$ KB. δ is a variable between 1 to 6 and represent the effect of quantizer step size in the frames size. *It is one of the factors that is changed for adaptation*. When there is no adaptation, $\delta = 6$ and corresponds to the highest quality of the MPEG video clip. In this case the I frames size are between 22.5 KB and 30 KB.
P frames are between $\delta \times 1.5$ and $\delta \times 2.5$. So the maximum allowable size of P frames is between 9 KB and 15 KB. B frames are between $\delta \times 0.5$ KB and $\delta \times 1$ KB and their maximum allowable size vary between 3 KB and 6 KB.(Figure 5-4)
As we see, the ratio of I/P/B frame's sizes is roughly 7/3/1 which is a reasonable ratio and comply with what is mentioned in very few available research done on MPEG video frame size [30], although this ratio is completely scene dependent and could be different for various video clips, but above ratio represent a typical value.

5.2.2 Routers

The nodes (routers) are output buffered datagram "store and forward" devices. They receive bits from the inputs (links) with a constant rate equal to the capacity of that link. As soon as the arriving bits make a complete packet, this packet is forwarded to the relevant output buffer based on its destination. The routers forward one complete packet to the relevant output buffer regardless of its size, i.e., all the inputs of the router are polled

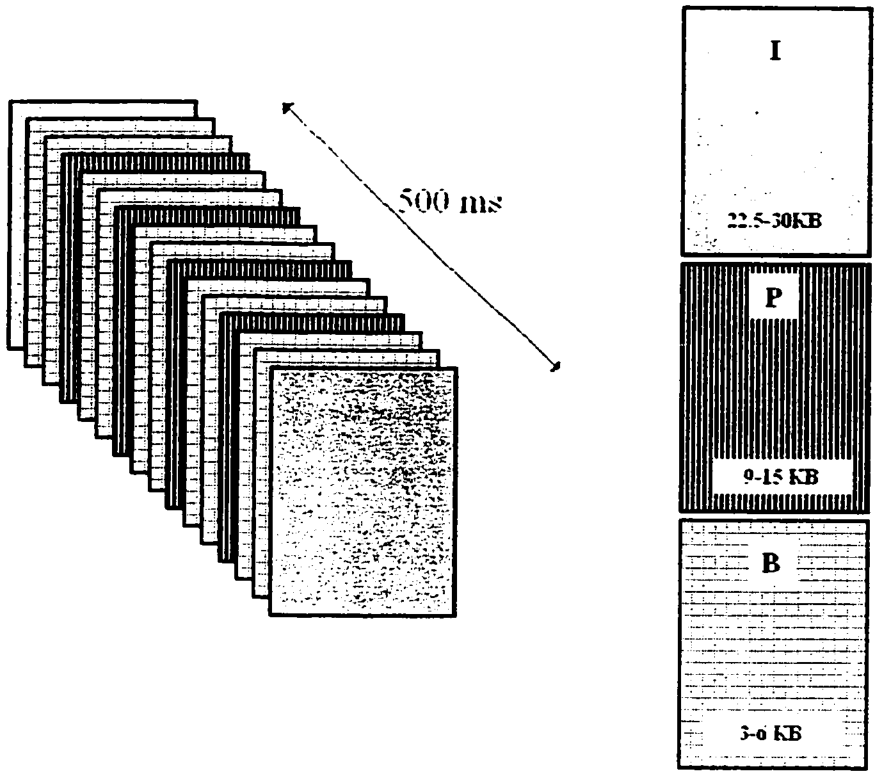


Figure 5-4: The simulation's GoP structure and frames' size

from different inputs. This is the least possible timing unit of the simulation.. The inputs of a router are treated equally. This means that there is no priority when the routers check their inputs. If the output buffer does not have enough space for a packet, the packet is discarded.

5.2.3 Dejitter buffers

We have simulated a dejitter buffer (see 2.1.2.1) at the receiver side. These buffers receive the packets from the last router and based on the presentation time, delivers them for decoding. The capacity of the dejitter buffer is limited in order to enable us to simulate buffer overflow.

5.2.4 Lost Packets

Like real world, a packet is considered lost when it faces one of the following situations:

- When dejitter buffer is full, the arriving packets will be dropped. This loss is called "loss due to buffer overflow".

decoding. The capacity of the dejitter buffer is limited in order to enable us to simulate buffer overflow.

5.2.4 Lost Packets

Like real world, a packet is considered lost when it faces one of the following situations:

- When dejitter buffer is full, the arriving packets will be dropped. This loss is called "loss due to buffer overflow".
- In the simulation, each packet carrying MPEG traffic has an attribute that represents Presentation Time Stamp (PTS) of MPEG information as described in 3.3. This complies with the packetization scheme that we follow in the simulation, which is putting Elementary Stream directly in RTP/UDP/IP packet as described in 4.1.

  The value representing PTS is equal to the summation of an acceptable constant delay for video clip and the time that the packet is generated. If the receiver receives a packet after this time, it is considered lost, as the time it should be presented on the screen has already passed. We call this "loss due to delay". The maximum (one way) allowable delay is 100ms in the simulation, which is in agree with what has been proposed in papers. [31].
- A packet may be dropped due to the router's output buffer overflow. We call this "loss due to congestion".

5.2.5 Links

The links connect the routers to each other and have certain capacity and provide certain propagation delay. The propagation delay depends on the length of the link. They transfer a constant stream of bits in each time unit, regardless of this bulk of bits is a part of a packet, or it includes more than one packet. This means that if we have a packet of 1000 bytes in one side of a link and capacity and propagation delay of this link is 300 KB/s and 4 ms respectively, during each ms (each iteration of **transfer()**), the packet at the sending end is "shrunken" by 300 bytes, but at the other side only after 4 ms the 300 bytes of information are passed to the input buffer of the router. The last 100 bytes of this packet will be accompanied with 200 bytes of the next packet at the fourth ms. We have assumed that the links are error free and does not introduce errors when transmitting the information. In our simulation links have length of between 300 km and 3000 km. Capacity of the links is one of the parameters that is changed in the simulation to verify its effect on the adaptation algorithm and packetization schemes.

5.2.6 Routing algorithm

Before transmission, at each source node, the "best rout" from the source to the destination that the packets should traverse, is calculated. In other words, source routing is used in the simulation. In source routing, the source is aware of the topology of the network and it assigns the path to each packet. So the router's job is reduced to examination of the header and put the packet in the appropriate output buffer.

By best rout we mean the rout that passing through it to reach the destination will result in a least possible delay. The delay a bit experience from source to destination, like real world, arises from three different sources:

1. Propagation delay of the links, forced by the nature, and is 3.3×10^{-6} sec/km in the simulation.
2. Queuing delay at routers: the routers are output buffered and the delay is due to the FIFO queuing in the output buffer before leaving the buffer. This delay is the result of the limited capacity of the links connected to the buffer, which is able to pass only a limited amount of bits every time unit.
3. Processing delay of each packet. This is the time that each packet experiences when its header is examined by the router. In the simulation each packet (regardless of its length) experiences the processing time.

The *average* total delay is used as the cost for calculating the least cost route ("best route"). At the beginning of the simulation, since there is no queue in the routers, the best rout is calculated just based on the propagation delay of the route. But after that, the average waiting time of the queues are added to the propagation delay and then the best routs are recalculated. Every one-second the best routs are recalculated and the routing tables are refreshed.

5.2.7 Source Fragmentation

After calculating the best routs for each source/destination pair, each packet knows that which rout it should pass to reach the destination. Based on the least allowable frame size that the packet will be faced during its travel through this rout, the Minimum Transmission Unit (MTU) of that rout is calculated. This means that if the route that a packet should pass consists of say two different layer two technologies with two maximum allowable packet size, the minimum packets size is selected as the minimum MTU of these two, and based on this size, the packets are fragmented to several packets. Above explanation is in compliance with the new IPV6 protocol, in which the fragmentation is done at the source. [32]

5.2.8 Error Propagation

Effect of MPEG video temporal and spatial error propagation as discussed in 3.2 is carefully considered in our simulation. For example loss of an I frame will cause all the Group Of Pictures (GoP) to be considered lost, or loss of the second P frame causes all dependent frames (totally 11 frames) to be lost and so on...Only B frames are "independent" and loss of them has no error effect on the other frames. If the lost packet contains only a part of a frame (a slice), then the error propagation have been considered appropriately as will be described in 5.2.11.2.1

5.2.9 Packet's attributes

The concept of object in programming languages like C++ that we used is very suitable for packets carrying the video information in the simulation. Each packet is represented as an object with certain attributes that most of them actually exist within RTP/UDP/IP headers, and some only are used by the simulation program. Some of these attributes are:
An ID no, Sequence no, Sender's and receiver's ID (addresses), size of the packet, Type of the packet (I, P, B or Data), the route it has passed so far, the time it should be presented, the delay is has experienced so far, is it fragmented or not? If yes, what is the sequence no of the fragment? Is it the last fragment? etc.

The simulation starts without adaptation. MPEG streams are sent with the maximum possible quality (hence maximum bit rate) for 26 seconds. Every two seconds a comparison between the transmitted sequence and what has been received is made for each user. This comparison calculates which packets have been lost and if the lost packet has propagated any syntactic/semantic error to the other packets, how much has

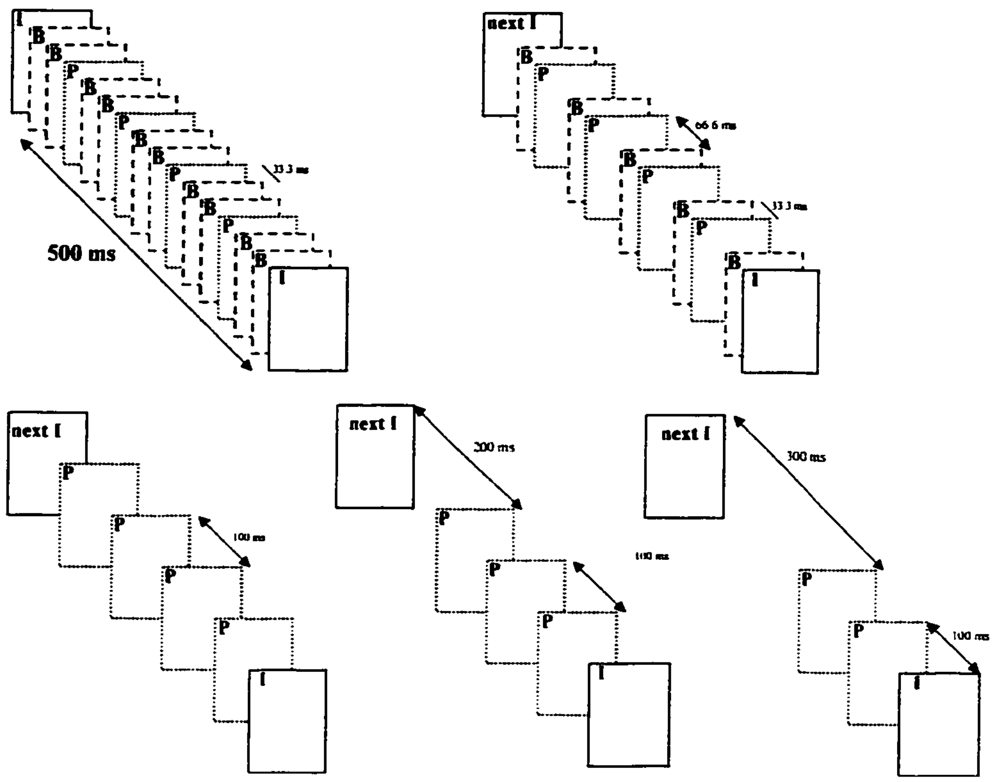


Figure 5-5: Frame Dropping Policy

been the average delay and delay variance of the receiving sequence, and how much has been the "share" of congestion, latency and receiver's buffer over flow as described in 5.2.4 in the lost packets so far. Each of these data and average of them are gathered. From the 26th second, adaptation is activated, i.e., based on the lost packets of each receiver, the relevant transmitter's data rate is changed by changing δ (as described in 5.2.1.2), and by dropping some frames as shown in table 2 below. Figure 5-6 shows how change of the parameters as shown in Table 1 leads to change of average bit arrival rate for each sender. Every two seconds a comparison is made and the sender's data rate is modified if been the average delay and delay variance of the receiving sequence, and how much has been the "share" of congestion, latency and receiver's buffer over flow as described in 5.2.4 in the lost packets so far. Each of these data and average of them are gathered. From the 26th second, adaptation is activated, i.e., based on the lost packets of each receiver, the relevant transmitter's data rate is changed by changing δ (as described in 5.2.1.2), and by dropping some frames as shown in table 2 below. Figure 5-6 shows how change of the parameters as shown in Table 1 leads to change of average bit arrival rate for each sender. Every two seconds a comparison is made and the sender's data rate is modified if necessary. The whole simulation takes 50 seconds, around half of it without adaptation and the other half with adaptation.

PLR	**GoP**	δ
PLR<0.02	**IBBPBBPBBPBBPBB**	**6**
0.02≤PLR<0.05	**IBBPBBPBBPBBPBB**	**5**
0.05≤PLR<0.08	**IBPBPBPBPB**	**6**
0.08≤PLR<0.12	**IBPBPBPBPB**	**5**
0.12≤PLR<0.15	**IPPPP**	**6**
0.15≤PLR<0.18	**IBPBPBPBPB**	**4**
0.18≤PLR<0.22	**IPPP**	**6**
0.22≤PLR<0.27	**IPPPP**	**5**
0.27≤PLR<0.33	**IBBPBBPBBPBBPBB**	**3**
0.33≤PLR<0.4	**IPPP**	**5**
0.4≤ PLR<0.45	**IBBPBBPBBPBBPBB**	**2**
0.45≤PLR<0.5	**IPP**	**4**
PLR≥0.5	**IPP**	**2**

Table 1: Details of the adaptation policy

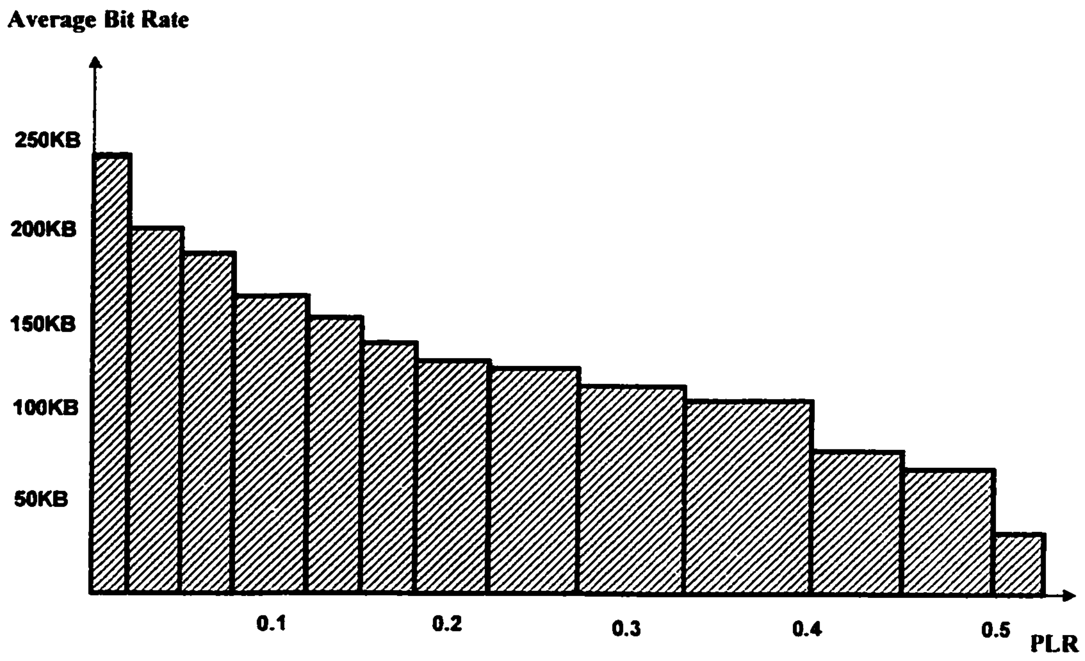


Figure 5-6: Chart presentation of average arrival bit rate versus PLR

entire frame into an IP packet. In this case, if an error occurs, the starting of the next frame will be the resynchronization point. We consider two possibilities both in agreement with the recommendation for packing of video ES as will be described below and simulate the two cases.

5.2.11.1 Packing a complete MPEG frame in a packet

In this case, we put one complete frame into one packet. This packet then would be fragmented if necessary as described in 5.2.7. So in this case each RTP/UDP/IP packet carries one fragment of a complete MPEG frame no matter exactly when this fragment in the frame starts and ends. We have assumed that our cluster of Internet consists of two different layer two technologies with one supporting packets of 3 times larger in size than the other (i.e., Token Ring with frame size of 4500 bytes and Ethernet with frame size of 1500 bytes). So basically our packets have two different sizes (except the last portion of a frame which may be put in a packet of smaller size).

different layer two technologies with one supporting packets of 3 times larger in size than the other (i.e., Token Ring with frame size of 4500 bytes and Ethernet with frame size of 1500 bytes). So basically our packets have two different sizes (except the last portion of a frame which may be put in a packet of smaller size).
For example an I frame of length 25000 Bytes which should pass through 4 different networks with the maximum allowable packet size of 4500,1500,4500 and 4500 bytes will be fragmented to 16 packet of size 1500 Bytes and one packet of size 1000 Bytes. Since these 17 packets belong to one frame, they are generated during 33 ms of time. So each packet is spaced roughly 33/17 ms with the other one in time domain, and so on...
Above assumption has implicitly assumed that our MPEG encoder is variable bit rate (VBR) as opposed to constant bit rate (CBR) encoders. In the first series of encoders, trying to maintain the same quality for all the frames leads to variable out put bit rates due to the fact that each frame has different size, but the frames are spaced equally in time. While in the second class of encoders (CBR), the effort is to maintain a constant bit rate output by changing the MPEG quantizer step size at the expense of different qualities of picture for different frames, as changing quantizer step size with each frame causes each frame to be quantized differently with the other, hence frames having different qualities.

5.2.11.1.1 Frame loss

An MPEG *frame* is assumed to be safely reached to its destination if *all* of its fragments have received by the receiver's buffer. Each fragment, if is not received by the destination after a certain threshold, will be dropped and is assumed lost. What we do is translation (or mapping) of latency to loss. So a packet will be considered lost if:
A- is lost due to congestion.
B- has a delay more than the threshold delay.
C- receiver's dejitter buffer is full and drops the packet.
A frame is lost if one or more of its fragments are lost due to a packet loss resulted from one of the above.

5.2.11.1.2 Average sequence delay

The receiver starts to present the frame after a certain amount of time (we call it "threshold time" which reflects PTS in the simulation). The application in use determines this amount (say 100ms one way for a typical videoconferencing). The decoder should decode the frames exactly at the same rate as they have been encoded in order to maintain the encoding quality. If we have had a 30 frames/sec sequence, the decoder also should decode each frame every 33.3 ms. The decoding time is based on the RTP timing stamp. In the simulation the decoding time is an attribute of each packet object.

We define the average network delay of a sequence as the average of the delay of its frames. The delay of a frame is equal to the delay of that fragment which has the longest delay. So if a frame has been fragmented to 10 packets and all of them are received by the receiver and delay of 9 of them is 90ms and one of them has a delay of 95 ms, the delay of the frame is 95 ms. This is the network delay, i.e., the difference between the time that a RTP/UDP/IP packet (a fragment of a frame) is delivered to the network and the time that the receiver buffer receives it. Note that if one of the fragments has a delay more than the threshold, this fragment is considered lost. In calculating the average delay, delay of the lost frame is considered to be equal to the "threshold time".

Calculation of the average delay of a sequence of frames is simply averaging the delay of frames that make that sequence. So our maximum delay is equal to the "threshold time" and corresponds to a sequence of pictures, which has lost all of its frames (At least one fragment from every frame). The receiver never receives this sequence. This makes the upper boundary of delay for a video sequence. The lower boundary of delay for a sequence is equal to the propagation delay plus processing time of the routers located in the least cost route. (i.e., when there is no queuing delay).

5.2.11.1.3 Delay variance of the sequence

Delay variance is simply the variance of the delay of the frames making the sequence. So the delay of each frame as discussed above is calculated. Then the average delay of the frames is calculated and finally the variance of the delay is calculated. The variance can give us an impression of how the receiving frames are spaced in time as will be discussed in jitter analysis later.

5.2.11.2 Packing a slice in a packet

In the second case, we put *each slice of a frame* in a RTP/UDP/IP packet. Recall that each frame consists of several (7 in QSIF standard) horizontal slices and each slice consists of eleven macro blocks. If one or more macro blocks belonging to a slice are lost, beginning of the next slice will be the new resynchronization point.

In this case one slice is put into one packet and this packet will be fragmented in the source node if necessary, the same way as described in 5.2.7.

5.2.11.2.1 Packet/Frame loss

This part of the simulation has been written carefully to reflect the real world's propagation of error in case of packet loss. A slice will be assumed corrupted if any of its fragments are lost or dropped due to increasing delay or buffer (dejitter or router's buffer) overflow. In case that the slice that holds the frame header is lost or corrupted, the whole frame will be assumed lost or corrupted. In the simulation we have assumed that the first slice of each frame has all the header information (which is a valid assumption) and loss of it leads to loss of whole frame. Also we have assumed that the first slice of an I frame holds the whole GoP header information and loss of it leads to corruption of whole GoP (again a valid assumption). In case that the slice does not have the header with it, loss of any fragment of it causes that the slice to be considered lost.

Note that in this case a corrupted slice will affect all slices that are made by applying motion vectors to its macro blocks (i.e., temporal error propagation). This means that we have assumed that when a slice of an anchor frame is lost, the effect of loss is propagated to the slice located the same place of the frames that have been constructed by applying prediction to that frame.

5.2.11.2.2 Average Delay of a sequence

Average delay of a sequence is averaging the delay of the frames that make this sequence. In this case delay of a frame is the delay of the slice that has the most delay and delay of a slice is equal to delay of the fragment that has the longest delay. So if all the fragments of a frame reach the destination safely, the fragment that has the longest delay will determine the delay of that frame. If a slice is not reached the receiver before the maximum acceptable delay, it is assumed lost and delay of the relevant frame is considered to be equal to "threshold time" ms. Note that in this case error concealment techniques can be used for the slice which has not been received (interpolation or repeat of adjacent slice for example) and the whole *frame* is not considered corrupted as the next slice is the resynchronization point, ***unless the header frame slice is lost.***

5.2.11.2.3 Delay variance of a sequence

As discussed previously, also in case of slice, based on the delay of frames, the variance of the frames making the video sequence are calculated by the program for further analysis.

5.3 Simulation Results

In this part we present the results obtained from simulation and discuss about them. The program calculates the *average* PLR, delay and delay variance of *the eight receivers*, and these averages are considered the results of the simulation.

Effects of changing different parameters on the proposed adaptation technique and a comparison between packing one frame or one slice inside one RTP/UDP/IP, both from network QoS and bandwidth efficiency point of views, will also be discussed.

5.3.1 Adaptation

Nearly all the curves show that at the first comparison, Packet Loss Ratio (PLR) is considerably decreased after applying the adaptation. Figure 5-7 and Figure 5-8 show PLR curves versus time for two cases with the parameters shown on them. These parameters, shown on all curves are as follows:

- RB (Router's Buffer size): Indicating output buffer size of the routers.
- DL (Data Length): Indicating size of the background traffic packets generated by the "non-MPEG" users.
- DR (Data Rate): Indicating load of the network.
- LC (LINK Capacity): Indicating capacity of the links connecting the routers.
- DB (Dejitter Buffer size): Size of the dejitter buffer, as described in 5.2.3.

Adaptation for both (and all other figures shown hereinafter) starts at the 26th second. Figure 5-7 and Figure 5-8 show PLR versus time when one slice and one frame is put in a RTP/UDP/IP packet respectively. (Shown as "slice" and "frame" respectively on the figures). Both figures show effectiveness of the adaptation: PLR considerably decreases after adaptation.

As the curves show PLR drops significantly 4 seconds after applying adaptation, i.e. at 30th seconds. The reason for this delay could be the time it takes for the buffers, which are full or nearly full before adaptation, to "flush" the packets.

As we see, there is a fluctuation in the PLR after adaptation. If we look at the chart of

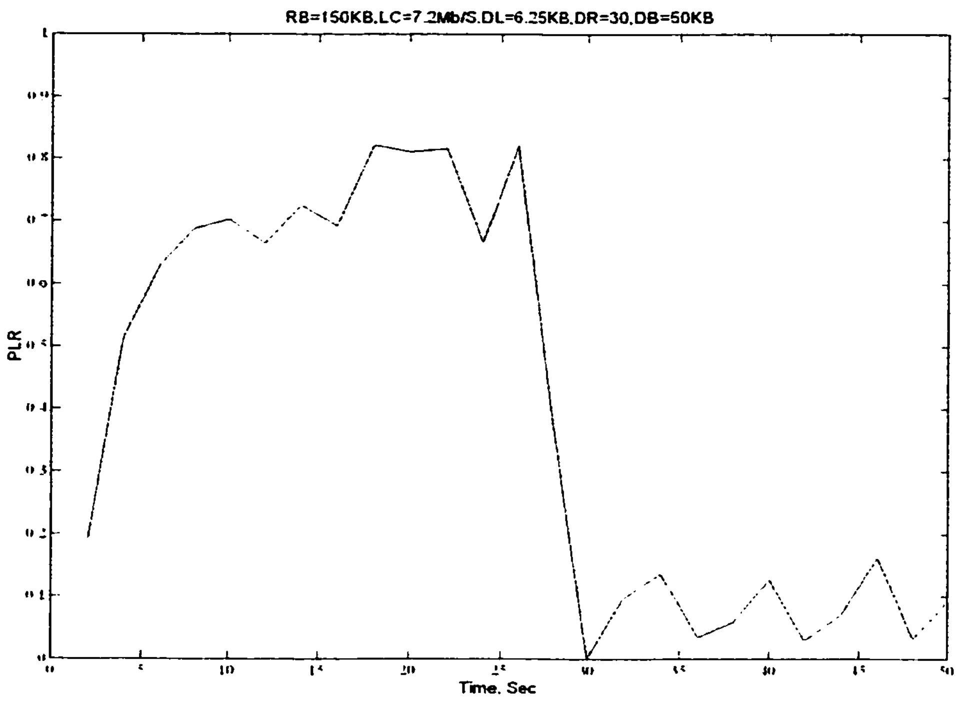


Figure 5-7: PLR vs. Time when one slice is put in a packet

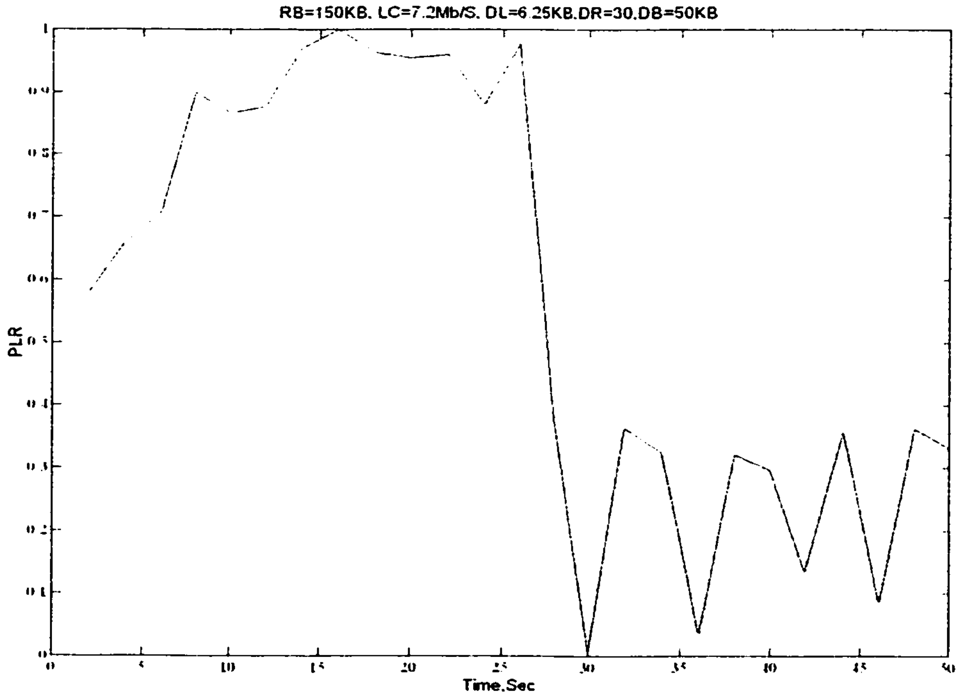


Figure 5-8: PLR vs. Time when one frame is put in a packet

Figure 5-6, we can understand the reason better: As PLR reduces to around zero; the generation rate is increased to its maximum by the adaptation. This increases the PLR, which in turn imposes decrease of the generation rate by the adaptation two seconds later at the time of recalculation of adaptation parameters; which in turn leads to measuring a lower PLR and increasing bit arrival rate two seconds later...

Note that measuring of PLR and applying suitable adaptation is done every two seconds and is not a continues process. These fluctuations could be reduced by adding more steps to the adaptation policy and lowering the difference between bit arrival rates of the adaptation steps. However, since we are working in a bursty environment, with uncontrolled and sudden arrival of background traffic, we should not expect to observe a completely flat PLR after the adaptation.

5.3.2 Effects of network's parameters on PLR curves

5.3.2.1 Effect of the network's load

Load of a network is characterized by how frequently the packets are sent to that network. This has been simulated through change of background traffic rate (parameter DR). Figure 5-9 and Figure 5-10 show the effect the network's load on the PLR when the slice and frame being put in a packet. (Note that the Link Capacity LC is different for the figures). Both figures show that the increase in load of the network as we expect, increases PLR. That is, more bursty background traffic will increase PLR.

Table 2 shows the effect of increasing the network's load for data packets of 6.25KB length, when each slice is put in a packet. The table shows that while still most of the packets are lost due to the delay (i.e., reaching the destination after the time that they should have been decoded), percentage of the packets that are lost due to congestion increases when the network's load is increased. This means that the more the generation rate of the background traffic the larger the probability of loss due to congestion.

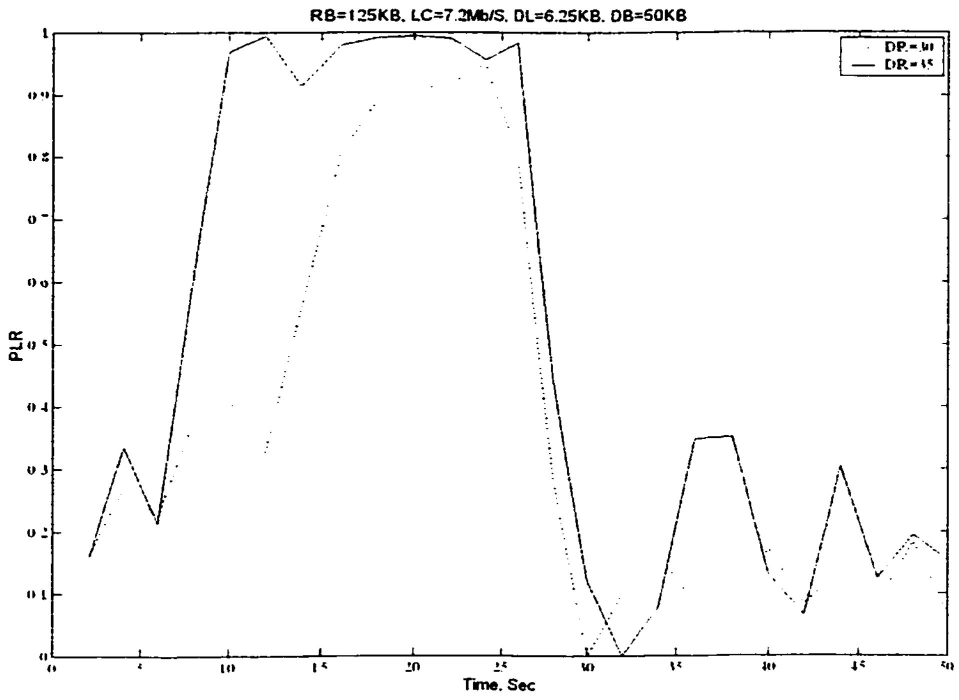


Figure 5-9: Effect of network's load on PLR (Slice)

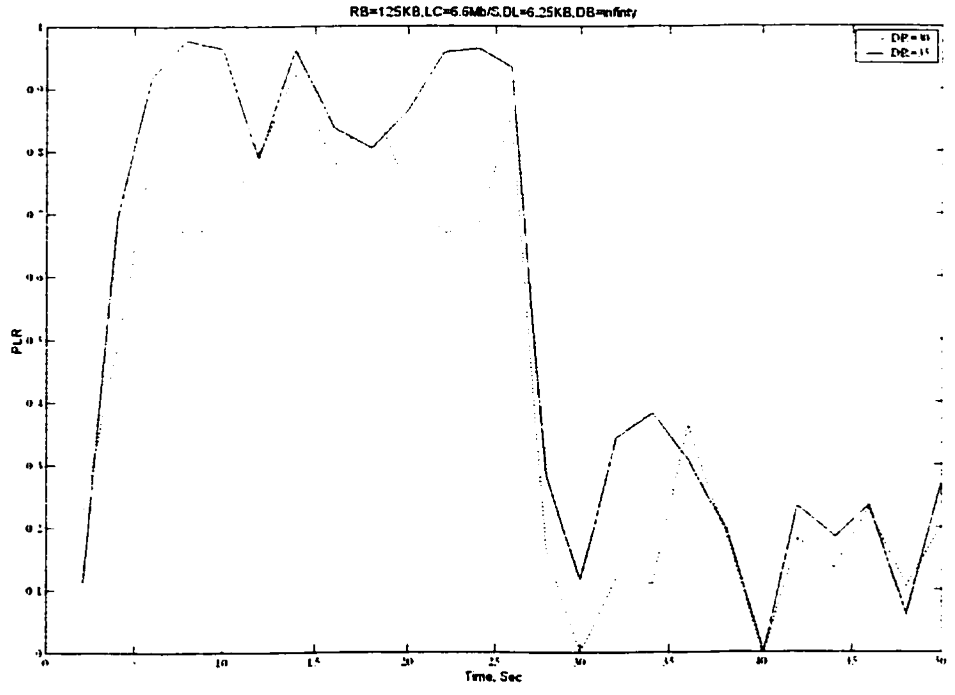


Figure 5-10: Effect of network's load on PLR (Frame)

Data Rate (DR)	Loss due to Delay	Loss due to Congestion	Loss due to Over Flow
30	89.2%	6.45%	4.2%
35	89.8%	8.93%	1.2%

Table 2: Effect of network's load on the probability of loss

5.3.2.2 Effect of background traffic packet's length

Figure 5-11 shows the effect of the length of the packets generated by the data traffic users (parameter DL) on the PLR curves for the case when a slice is put in an RTP/UDP/IP packet. Figure 5-12 is the same for the case when a complete frame is packetized in a packet before transmission through the network. Both figures based on three different run of simulations with three different packet sizes of data traffic (3.75KB, 6.25KB and 8.75KB). As we see PLR increases (before and after adaptation) when the background traffic packet's length is increased. The adaptation is less effective when the traffic packet size is larger for both cases. The reason is that the bigger the size of the background traffic packets, the less the share of MPEG users in the total traffic. Since adaptation has no control over the background traffic, so the change of bit arrival rate can only be effective when the MPEG traffic is comparable to the amount of the data traffic. So in a best effort network, an adaptation scheme does not work well if the network carries a huge amount of background traffic and only reservation (RSVP), priority mechanisms (Diff-Serv) can be useful in such networks for delivering real-time multimedia.

Based on the figures, when the background traffic packet lengths are large enough there is no difference between putting a slice of a frame or a complete frame in a packet. In both cases nearly all video data have been lost before adaptation. A detailed comparison between frame and slice will be given later in 5.3.3.

Based on the simulation results, when the length of the packets becomes larger, the percentage of lost packets due to congestion increases from 6.5% to 11.5%. This means the shorter the length of the background traffic, the less the probability of loss due to congestion. This is due to the explosive nature of the data traffic. So when the data traffic packets are generated with longer lengths, the output buffer of the router is filled with a larger packet, leaving less room for the video packets that will arrive later. On the other hand since more packets are lost due to congestion, less packets reach the destination, decreasing share of the packet loss due to receiver's buffer overflow from 4.2% to 0.05%. (See Table 3 below)

DL	Loss due to Delay	Loss due to Congestion	Loss due to Over Flow
6.25KB	89.8%	6.4%	4.2%
8.75KB	87.8%	11.5%	0.59%

Table 3: Effect of DL on the loss category

As we see most of the packets are lost due to latency, which implies that increasing the capacity of the link should enhance the results considerably. This will be confirmed by the simulation results as will be discussed later.

5.3.2.3 Effect of Link Capacity

Figure 5-13 and Figure 5-14 show how the PLR changes with different capacity of the links for slice and frame being packed in each RTP packet respectively. As one expects and the results show, increasing the capacity of the links has a considerable impact on PLR curves.

Increasing the capacity of the links have two simultaneous and related effects on the loss: It decreases the probability of buffer overflow at the router's as the larger the capacity of the link the buffer is emptied sooner, and at the same time it shorten the time that a packet experiences in a buffer, hence decreasing the probability of loss due to latency. The results generally confirm the above conclusion, although the loss due to congestion seems does not completely follow it. The results show a considerable improvement in the PLR curve of course by increasing link capacity, while the main cause of loss still remains latency.

Table 4 shows how the share of each category in packet loss changes with the link capacity in the case of a frame being packed in a packet.

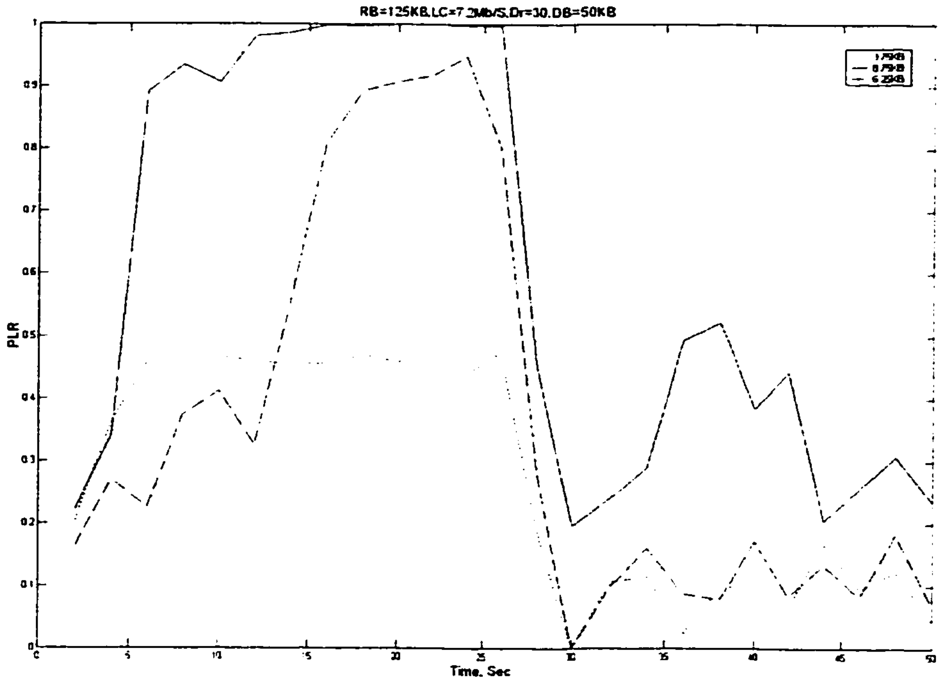


Figure 5-11: Effect of data traffic length on PLR (Slice)

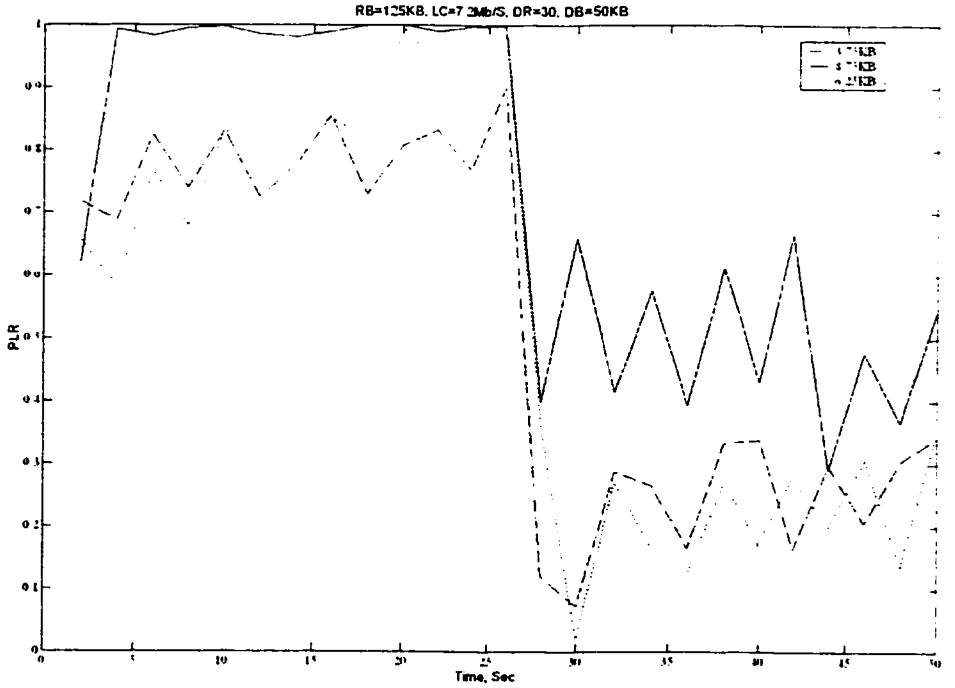


Figure 5-12: Effect of data traffic length on PLR (Frame)

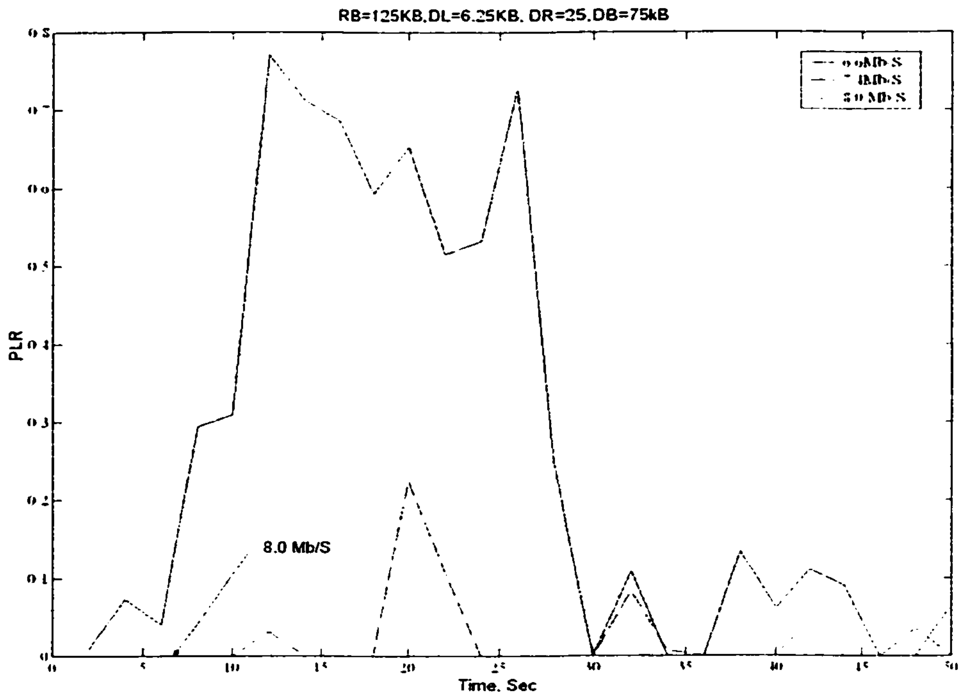


Figure 5-13: Effect of link capacity on PLR (Slice)

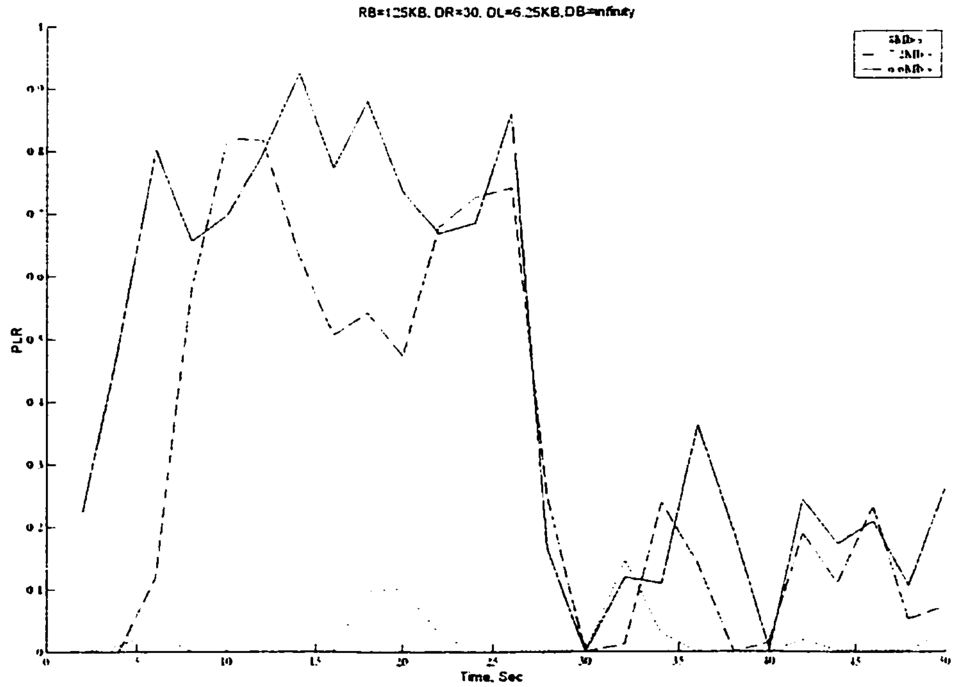


Figure 5-14: Effect of link capacity on PLR (Frame)

Link capacity-MB/S	Loss (delay)	Loss (congestion)	Loss (overflow)	Total lost packets
6.6	90.6%	8.1%	1.3%	44717
7.2	74.2%	13.8%	2%	31425
8	71.4%	12.3%	16.3%	3612

Table 4: Effect of increasing Link capacity to the packet loss categories (frame)

5.3.2.4 Effect of capacity of buffers of the routers in the PLR

What happens to PLR when we increase capacity of routers? This question is not very easy to answer. We expect that the probability of loss due to congestion drop as the buffer capacity increases. But on the other hand these newly arrived packets that have successfully passed the danger of congestion join a larger queue and should spend more time in that router to be passed to the next router. That is why when we look at Figure 5-15 that shows effect of increase in buffer of routers, it is difficult to find a rule to describe it. The dot curve belongs to the case when the capacity of the buffer is 125KB and solid curve shows PLR for the same conditions with buffer capacity of 175KB.

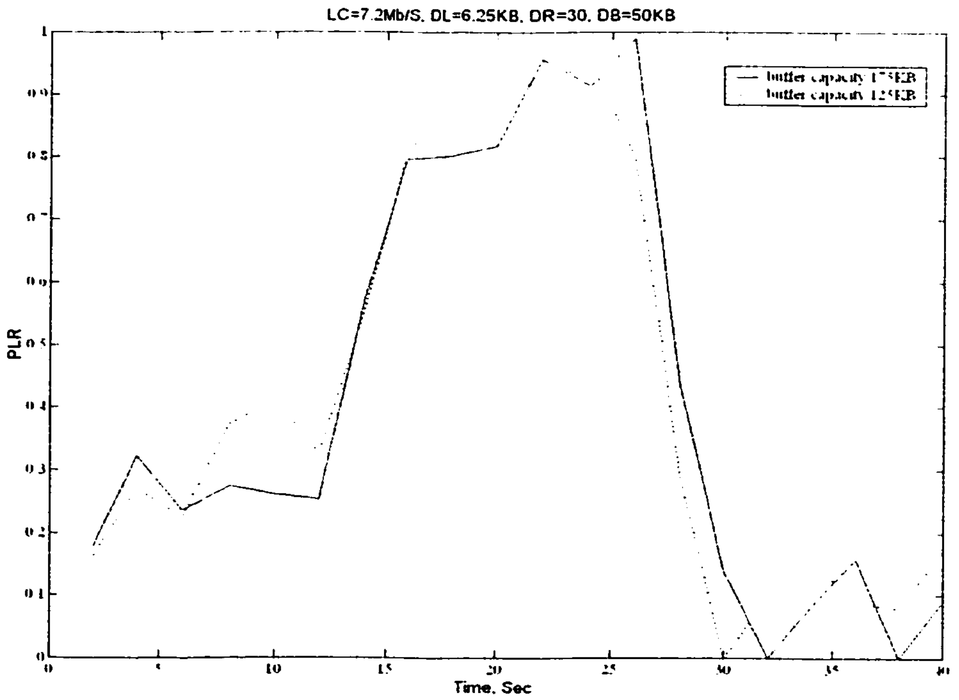


Figure 5-15 : Effect of router's buffer capacity on PLR

On the other hand as the following table shows the percentage of loss due to congestion decreases from 6.4% to 4.3%, while still the dominant reason for loss remains delay.

Buffer Capacity	Loss due to Delay	Loss due to Congestion	Loss due to Over Flow
125KB	89.2%	6.45%	4.2%
175KB	91.4%	4.3%	4.3%

Table 5: Effect of Router's buffer capacity on loss categories

5.3.2.5 Effect of Dejitter buffer

Figure 5-16 show the effect of dejitter or smoothing buffer's capacity on the PLR for Slice mode. As expected the PLR decreases with the increase in the buffer capacity. The reason is that increasing the capacity decreases the probability of loss due to dejitter buffer over flow.

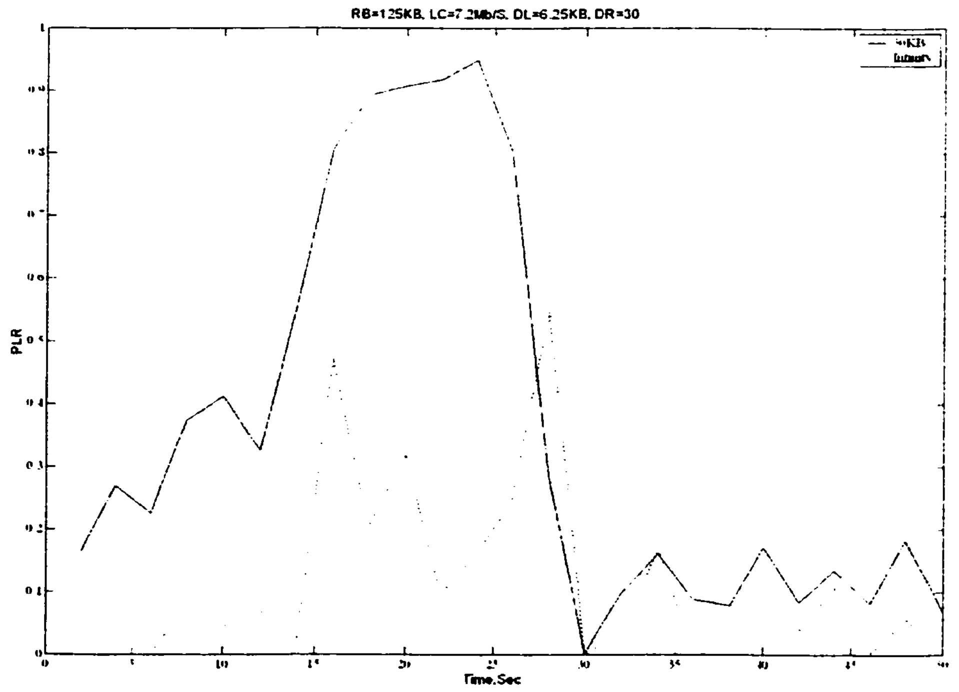


Figure 5-16: Effect of Dejitter buffer capacity on PLR (Slice)

Figure 5-16: Effect of Dejitter buffer capacity on PLR (Slice)

5.3.3 Comparison between RTP/UDP/IP packet sizes

As described previously, based on RFC2205 recommendations, one or several slices (i.e., an integral number of slices) may be put in an IP packet. Based on QSIF standard, a frame is composed of seven slices. (Figure 5-17). We have used this standard in the simulation and assumed that a frame has seven slices with *equal sizes*. The latter assumption is only for simplicity of simulation, as the size of a slice, depends on the information it carries and how fast the information inside the macroblocks making that slice changes compared to previous macroblocks.

Figure 5-18 shows a comparison between putting one complete frame or one slice in each RTP packet. The figure illustrates PLR curves for six different conditions of the network as indicated on each curve. The results show that the packet loss ratio is less if we put each slice in a packet. This can be justified as below:

When our smallest video entity is a frame, one frame (or seven slices in QSIF) is put in a RTP/UDP/IP packet, and this packet is fragmented based on the route it will pass (either "source fragmentation", which is the case in our simulation or fragmentation as it passes the route). Since when a fragment of a packet is lost, normally the whole packet is considered useless by the IP, this packet will be discarded and will not be delivered to upper layers for decoding. In this case, the next resynchronization point will be the beginning of the next frame as we saw in chapter two. On the other hand if the frame in error (i.e., the frame that has lost at least one of its fragments) is an I or P frame, then the effect of this lost frame will be reflected on the other frames (temporal error propagation).

In the case of one slice being put in a packet, again the loss of any fragment of the slice causes the whole slice to be lost, but the resynchronization point in this case will be the start of next slice. So loss of a fragment of a slice is only spatially propagated through that slice till the next slice. Temporal error propagation in this case is very difficult to determine exactly. For the simplicity, we assumed that a defected slice of an anchor frame defects the slice in the same location of the frames that use that anchor frame for decoding.

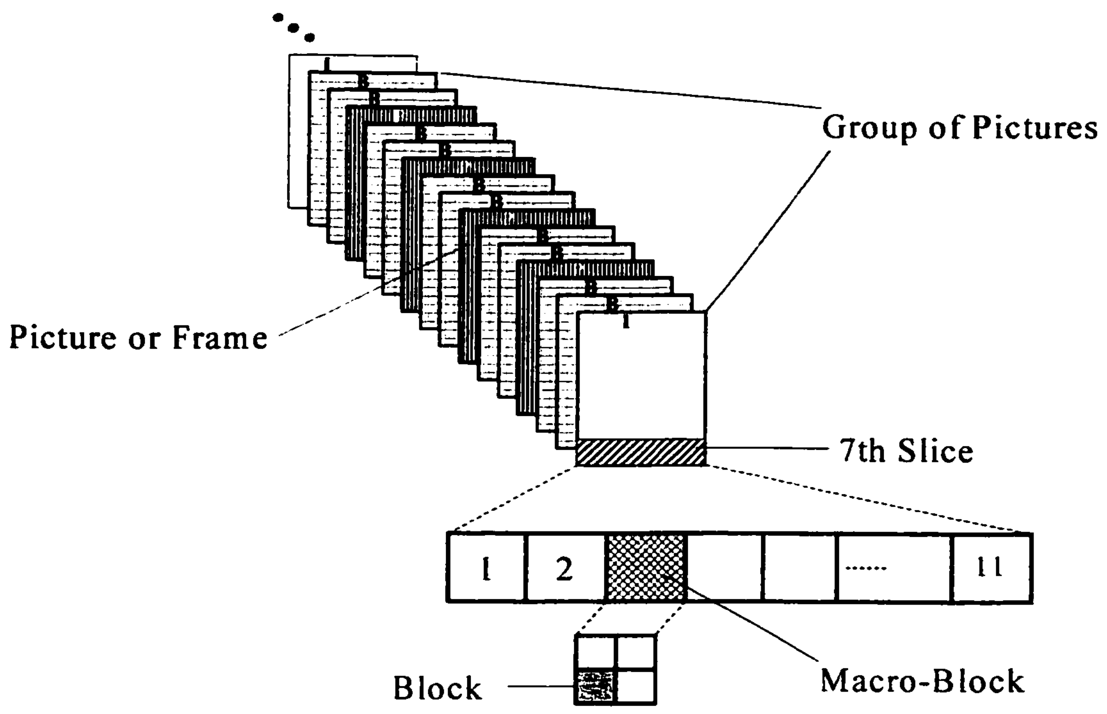


Figure 5-17: MPEG hierarchy in QSIF standard

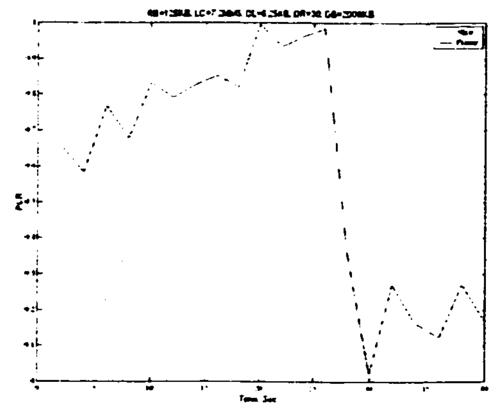
Frame for example causes the whole GoP to be lost, while loss of the 2nd slice of I frame will be propagated to the other frames as described above.

To be able to compare the results, consider an I frame with average size of around 26KB. If the minimum frame size that this frame will see in its route is 1500B (i.e., Ethernet), then this frame is fragmented to around 18 fragments of 1500B (disregarding header of MTU). Loss of each of these fragment leads to loss of the whole frame (and considering temporal error propagation) to loss of a whole GoP. Now if we put one slice of this frame in a packet, assuming the average size of each slice being equal, each slice will have a length of around 4000B, and each slice is fragmented to 3 fragments leading to a total of 21 fragments. Loss of the first three fragments of this frame (which make the first slice, and the first slice having the frame header information) have the same effect of loss of one fragment in case we put a whole frame in an IP packet (i.e., loosing whole GoP), but the loss of any other fragment will cause the relevant slice to be lost, which in case of I frame will be propagated till next 15 frames causing loss of total 15 slices. Furthermore the fragmented packets are more likely to be lost as obviously the probability of loss will be propagated till next 15 frames causing loss of total 15 slices. Furthermore the fragmented packets are more likely to be lost as obviously the probability of loss increases when the number of fragments that make a packet increases. The simulation in each run tests if a lost packet had been fragmented or not. The following table presents the number of the packets that have been lost and had been fragmented in slice and frame modes with the parameters of the simulation are the same constant for both cases.

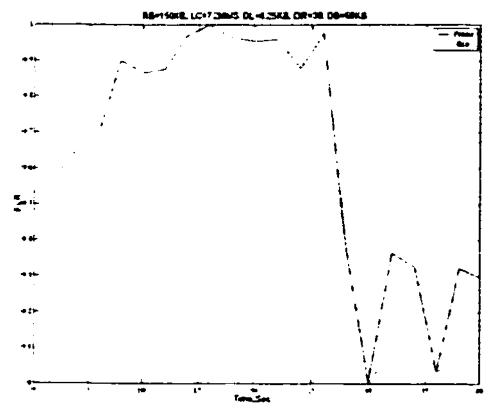
	RB-125KB, DR-30, DL-6.25KB, LC-7.2Mb/s36, DB-50KB	RB-125KB,DR-35,DL-6.25 KB,LC-7.2Mb/s36, DB-50KB	RB-150KB, DR-30, DL-6.25KB, LC-7.2Mb/s36, DB-50KB	RB-175KB, DR-30, DL-6.25KB, LC-7.2Mb/s, DB-50KB
Frame	25247	50282	33316	57407
Slice	9890	25616	12202	16637

Table 6: No of fragmented packets for simulations running with different parameters

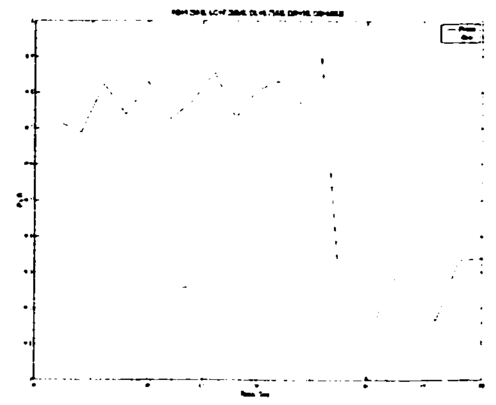
As the table shows, number of fragmented slices is more for a sequence of video when each frame is put in the RTP/UDP/IP header, thus increasing the probability of loss. Another point, which should be noted about the differences of putting slice or frame in an IP packet, is the "more explosive" nature of packets when we put a frame in a packet. Assume above case when an I frame is put in an RTP packet and is fragmented and delivered to the network. In this case, for every $\frac{33}{17} = 1.94$ ms, a packet of 1500B, which is a fragment of the I frame, is fed to the network. For the slice mode, the story is more or less the same: for every $\frac{33}{21} = 1.57$ ms a packet (a little bit smaller than) 1500 B is fed to the network. But consider a case when our frame is a B one with a size of 4500 B. In the case of a complete frame being encapsulated, we will have three fragments of 1.500 B fed into network each 11 ms. Now if we put each slice in a packet, every $\frac{33}{7} = 4.71$ ms we will have a packet of 640B. It means that in the case of frame, we have bigger packets, arriving in the network less frequently, whereas in slice mode, we have smaller packets arriving to the network more frequently. While our links "sink" a continuous stream of bits, a sudden arrival of bits will increase the probability of congestion



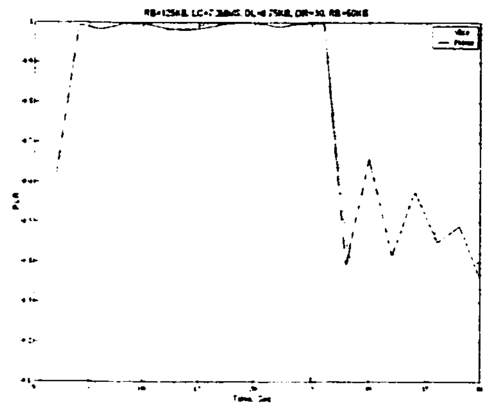
(a)



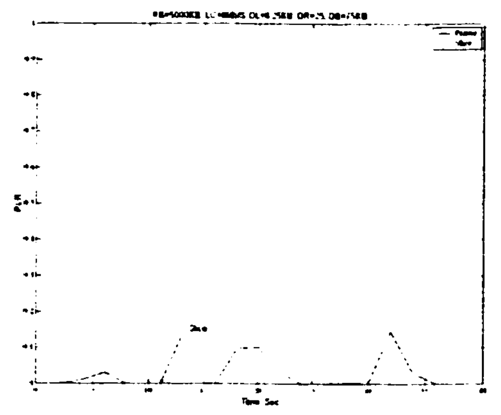
(b)



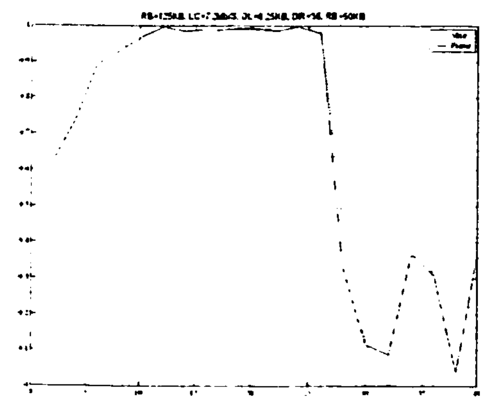
(c)



(d)



(e)



(f)

Figure 5-18: Comparison of PLR for Frame and Slice .The six sub-plots correspond to six different network's condition as indicated on them. Dash and solid curves represent one slice and one complete frame being encapsulated in one packet respectively. In all curves adaptation starts at the 26th seconds.

5.3.4 Bandwidth Efficiency

The price that should be paid for putting each slice in a RTP packet is transmitting more "useless" information due to more frequently transmission of headers.

So far we did not consider packet headers and assumed that our packets are pure data. In fact each RTP/UDP/IP packet has a header of 40 bytes length that does not carry any useful information from the receiver side point of view. To this 40 bytes, 4 bytes of MPEG information header as specified in RFC2250 should be added, which makes a total 44 bytes header for data payload. (Figure 5-19)

IP Header 20 Bytes	**UDP Header 8 Bytes**	**RTP Header 12 Bytes**	**Mpeg Header 4 Bytes**	**Data Payload**

Figure 5-19: Headers for carrying MPEG video Elementary Stream

When we limit ourselves to put one slice in a packet regardless of its size, this packet may carry a small amount of data, especially when B frames are considered. The small size of the data payload leads to inefficiency

We define efficiency as:

$$\eta = \frac{data}{data + header}$$

where data and header are the length of the data payload and the length of the header of the packet respectively. The optimal efficiency that could be obtained is given by:

$$\eta_{opt} = \frac{MTU - header}{MTU} = 1 - \frac{header}{MTU}$$

which corresponds to a payload filing the MTU completely. If the payload does not fill the MTU completely, optimal efficiency η_{opt} will be decreased by

$$\frac{data - S_{non-opt}}{MTU}$$

where $S_{non-opt}$ is the size of a non-optimal packet. So we will have:

$$\eta_{non-opt} = \eta_{opt} - \frac{data - S_{non-opt}}{MTU} = 1 - (\frac{header}{MTU} + \frac{data - S_{non-opt}}{MTU}) = \frac{S_{non-opt}}{MTU}$$

In the case of Ethernet as the predominant MTU at internet with 1500 bytes length (1460 bytes of data and 40 bytes the Ethernet header) the optimal efficiency is reached whenever the payload is 1416 bytes of length, so that with the addition 44 bytes of RTP/UDP/IP header for carrying MPEG video, it could be placed in one Ethernet frame. In this case $\eta_{opt} = \frac{1416}{1500} = 0.944$ or 94.4%.

For Ethernet the non-optimal efficiency is given by:

$$\eta_{non-opt} = \frac{S_{non-opt}}{MTU} = \frac{S_{non-opt}}{1500}$$

In the case of packing a complete frame in a packet, this does not cause any problem, as frames are normally larger than 1416 bytes. But when putting each slice in a packet, the problem shows itself.

Assume that the information in a frame has been uniformly distributed among the slices, i.e., each slice of a frame carries the same amount of information. In this case, when using QSIF (in which each frame consists of seven slices), for transmitting the information over Ethernet, a frame should have the size of around 10KB ($7 \times 1416 \approx 10000B$), for the packing to be optimum. For a typical B frame having the size of around a couple of KBs, packing each slice in a RTP/UDP/IP packet will lead to inefficiency. The problem will show itself more when we consider the fact that B frames are the most frequently transmitting frames in a typical video sequence.

Let's examine the case for a real world sequence of frames measured in [22]. The average size of B frames for a short CNN video clip containing news and commercial was found to be around 3020 bytes/frame for SIF standard (352x240 pixels), which, assuming uniform distribution of information in a frame, is around $\frac{3020}{11} = 274$ bytes/slice. For the Ethernet case the efficiency will be around $\frac{274}{1500}$ or 18.2% which is 76% less than optimal packing!

If we pack a complete B frame into a RTP/UDP/IP packet then 2832 bytes of this frame will be sent in two (Ethernet) frames and the remaining 20 bytes in the next packet and totally 132 (44x3) "non-useful" bytes of the header information are sent with the "useful" the payload information.

For the case of slice, eleven packets each containing one slice are sent, hence 484 (44x11) bytes of header are transmitted; an increase of 352 bytes for 3020 bytes information that is a huge waste of bandwidth which, is the drawback of putting one slice in one packet.

Figure 5-20 shows total header's traffic that will be injected to the network versus size of MPEG frames for both frame and slice cases in SIF standard. As it is shown, for smaller MPEG frames, putting a complete frame inside a RTP/UDP/IP packet and transmitting it introduces less header traffic, hence increasing efficiency, while in the case of encapsulating each slice in a RTP/UDP/IP packet, the total header traffic is independent of frame size and after a threshold is doubled. The figures are valid when the packets are sent through Ethernet. We have also assumed that the size of MPEG frames has been distributed uniformly over the slices

Figure 5-21 shows how the (non-optimal) efficiency changes with the MPEG frame size for the case when IP packets are sent through Ethernet frames. The curves belong to SIF and QSIF standard assuming that we have put one slice in each RTP/UDP/IP packet. Again we have assumed that the size of MPEG frames has been distributed uniformly over the slices. As we see from the figures for SIF standard, if the MPEG frame has size of less than around 16KB, the efficiency will be less than optimal. For QSIF case, as we showed before, this value is around 10 KB.

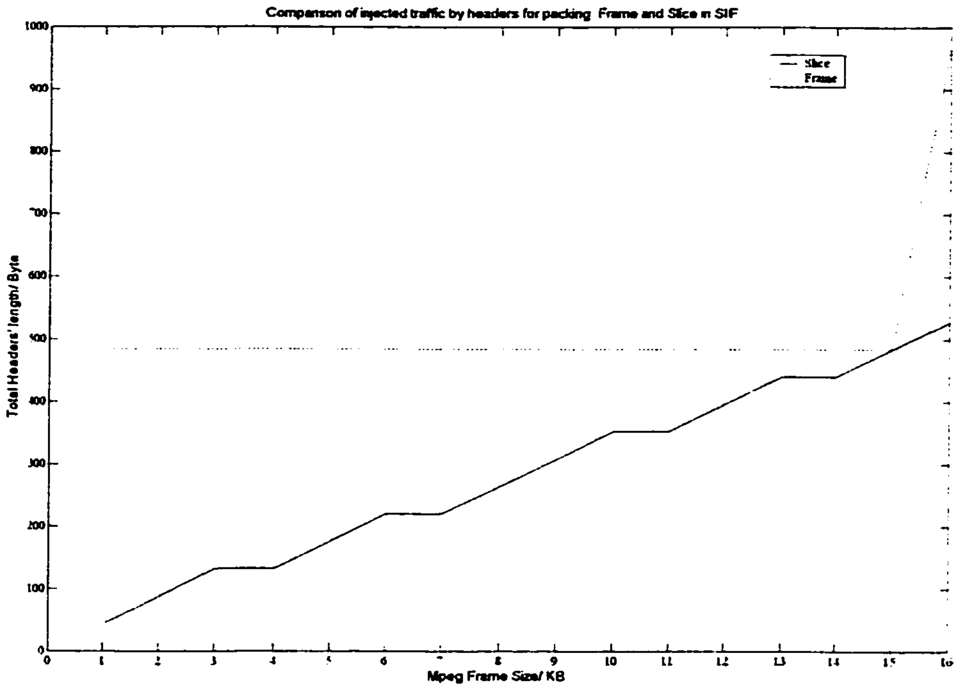


Figure 5-20: Total headers' bytes vs. MPEG frame size in SIF standard

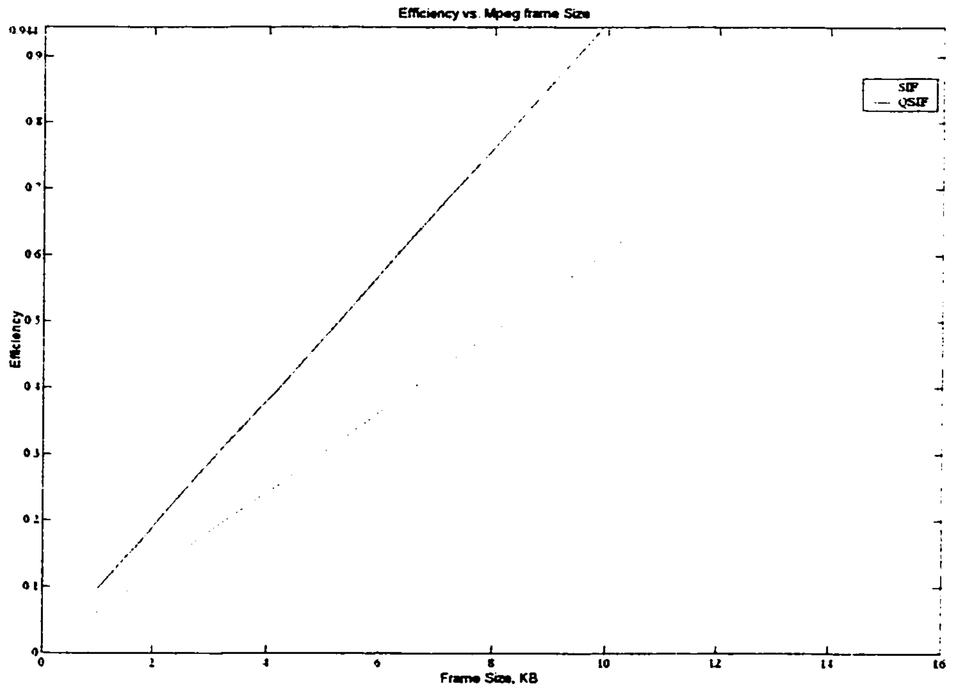


Figure 5-21: Efficiency vs. MPEG video frame size for QSIF and SIF standard

5.3.5 Delay

As mentioned previously, the simulation also calculates the mean delay of each sequence of MPEG video, i.e., the average delay that frames of sequence experiences. Bearing in mind that the main reason for the packet loss as we showed previously was due to unacceptable delays, one should expect a similarity between curves of delay and PLR. In fact if we compare Figure 5-22, which shows the delay for a typical case, and its corresponding PLR shown in Figure 5-23, we see a strong similarity between shapes of the two curves. We remind that we considered the maximum delay (100ms) for the lost frames.

For a case that PLR was zero (due to the large capacity that we allocated to the link, dejiter and router's buffer parameters of the simulation program in a not very "unordinary background traffic conditions and corresponds to the dot curve of Figure 5-13), the delay curve is shown below in Figure 5-24. Actually we still see a similarity in shapes in this particular case too.

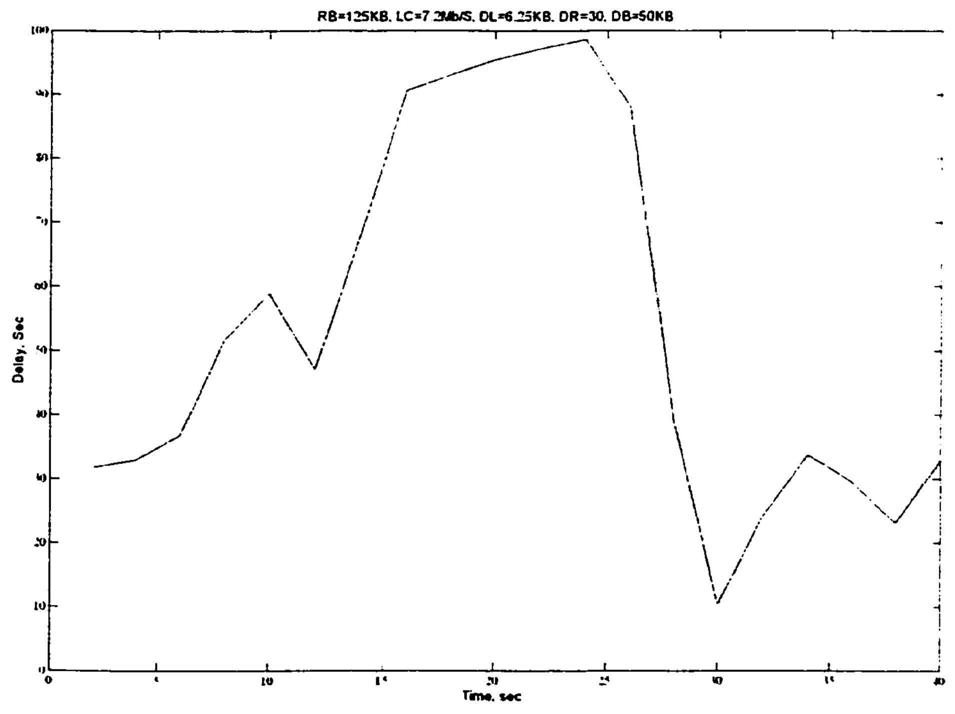


Figure 5-22: Corresponding delay curve of Figure 5-23

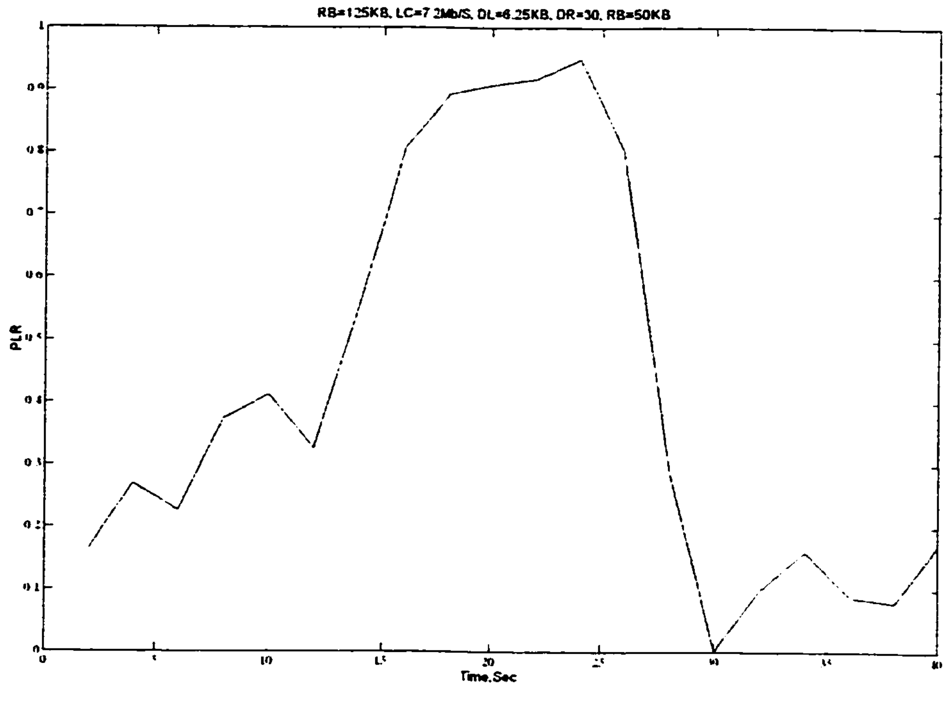


Figure 5-23: Corresponding PLR of Figure 5-22

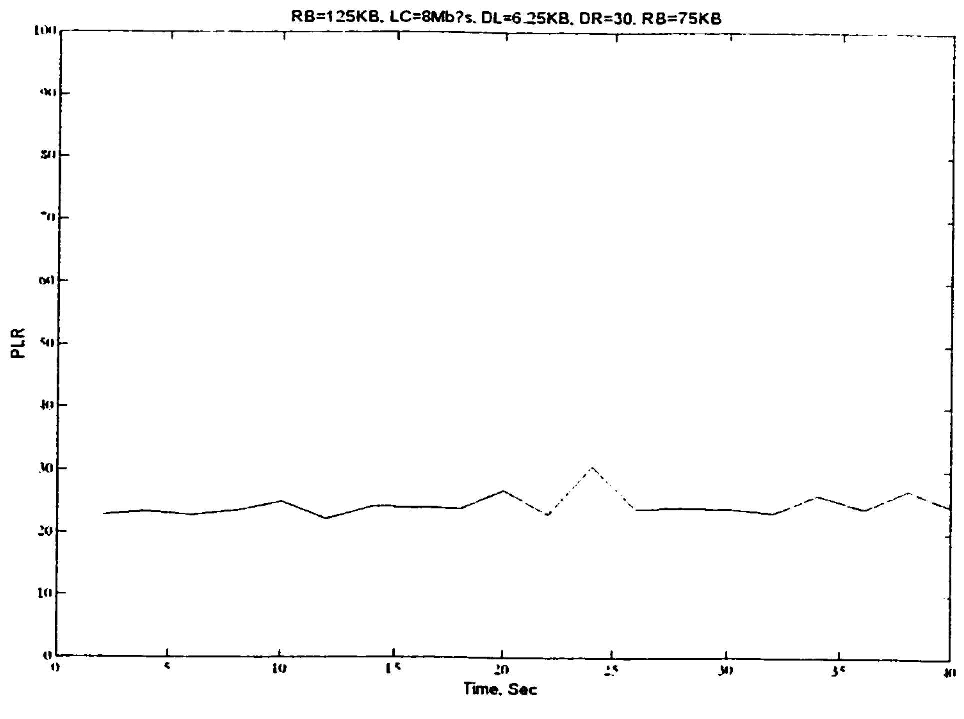


Figure 5-24: Delay curve for the case when PLR equals zero

5.3.6 Variance of delay

One of the important quality measures of real time applications in packet switched networks is jitter [33].

Jitter is defined as variations in delay and normally delay variance is considered to represent it. As in some books propose other methods for jitter measurements [34], we simply call this measurement as delay variance.

The reason for jitter is *mainly* due to queuing delays. In other words, since propagation and processing delays are constant, any variation in average delay is because of the variation in the time that packets spend in the queues. We mentioned "mainly" because the variation is also a function of the routs the packets pass from the source to the destination specially in our simulation that the routs are refreshed frequently, but since there is not a significant difference in propagation delay of different paths, this effect is less important.

The time that a packet spends in a queue is a parameter of the traffic, and the variation of this time depends on how much the traffic affects the packets carrying real time applications.

From above we expect that the variance of delay increase in an environment with a more bursty background traffic. In fact, Figure 5-25 confirms this. The figure shows delay variance for a case that the rate of arrival background traffic (network's load) has been changed. The curve shown in dot corresponds to the situation when the rate has been changed from 25 to 35 while all other parameters are unchanged. Figure 5-26 show another case when the length of the internet traffic packets has increased from 3.75KB to 6.25 KB. As we see again in average the variance has increased, as the arrival of larger packets in add to variation in delays. The increase in the variance is not as clear as it was in the previous figure.

We see a jump in the variance in Figure 5-25 after adaptation. This case corresponds to a highly loaded network (DR=35), which provides a high PLR before adaptation, mainly

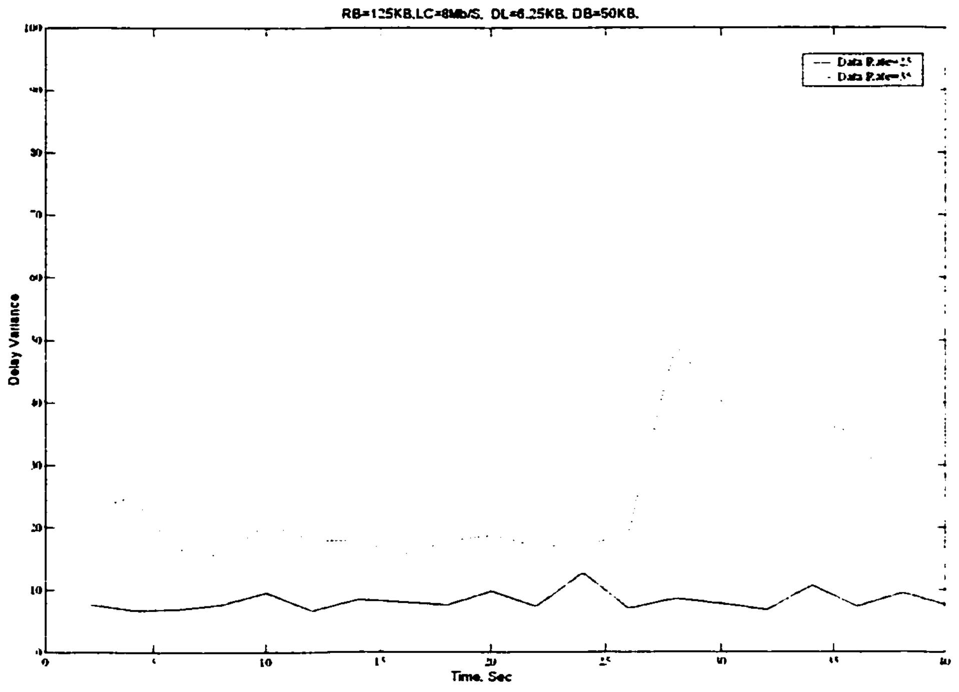


Figure 5-25: Effect of network's load in delay variance for MPEG video

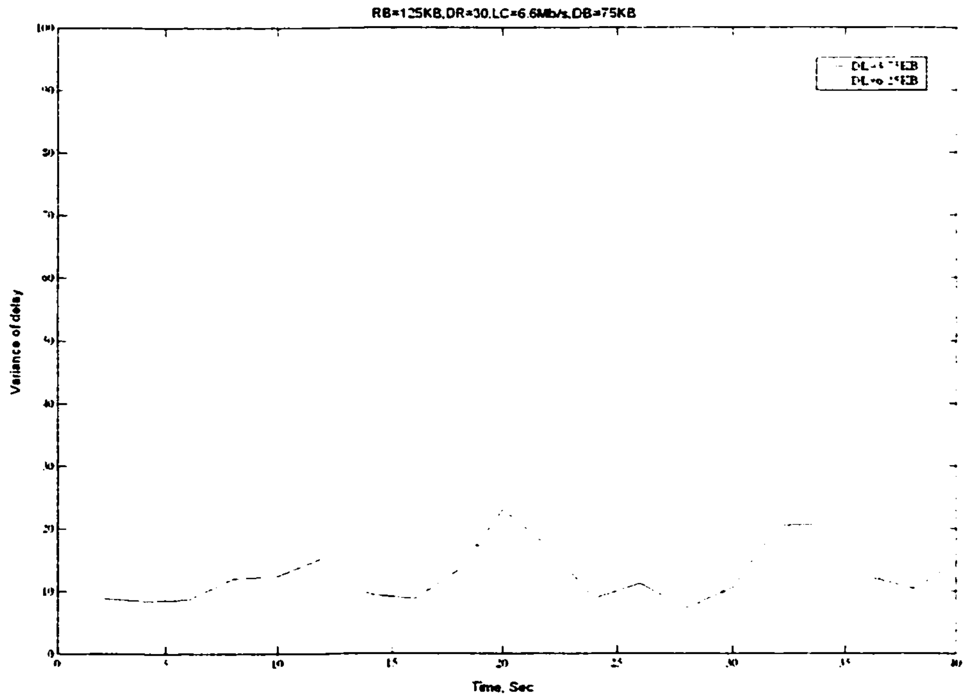


Figure 5-26: Effect of Background traffic packet length on the variance

due to delay of packets as we saw previously. In this case average delay of the sequence tends to be around 100 ms (since it is very probable that one frame of the sequence to be considered lost due to loss of one slice. Look at 5.2.11.2.2). When a sequence has average delay of 100 ms or something close to this value for a relatively long period of time, obviously the variance will be low. After the adaptation, however, the delay of the sequence is *not* pushed to stick to 100ms by the lost slices as it did before, because at this time loss is less. In this case, the delay can be anything resulting from the network's condition and the variance shows a higher value.

5.3.7 Increase in the maximum acceptable delay

The maximum acceptable delay cannot be changed, but for the purpose of investigation, assume that our application allows that the maximum one-way acceptable delay to be 200ms (instead of 100ms that we have assumed so far). In this case PLR shows an improvement as depicted in the following figures. In Figure 5-27 what we see is a shift of around 10 Seconds in the PLR curve to the right. This could be the time that it takes for the buffers of the routers to be occupied "enough" to introduce queuing delay more than the new acceptable delay. So it seems that the story would be the same after this shift in time, i.e., again the delay will cause the packets to reach the destination late. Note that in Figure 5-27 we are in a rather high traffic environment (DR=35), where the background traffic has a considerable share in occupying the routers buffers, hence adding to the delay and congestion, while in Figure 5-28 we are in a low traffic environment (DR=25), and as we can see prolongation of acceptable delay has an obvious effect in the reduction of PLR.

Table 7 shows how number of packets and share of them in loss has changed with the change of maximum acceptable delay for both cases during the 26 seconds before adaptation. For figure 5-25 we see that number of the lost packets are considerably less, because of

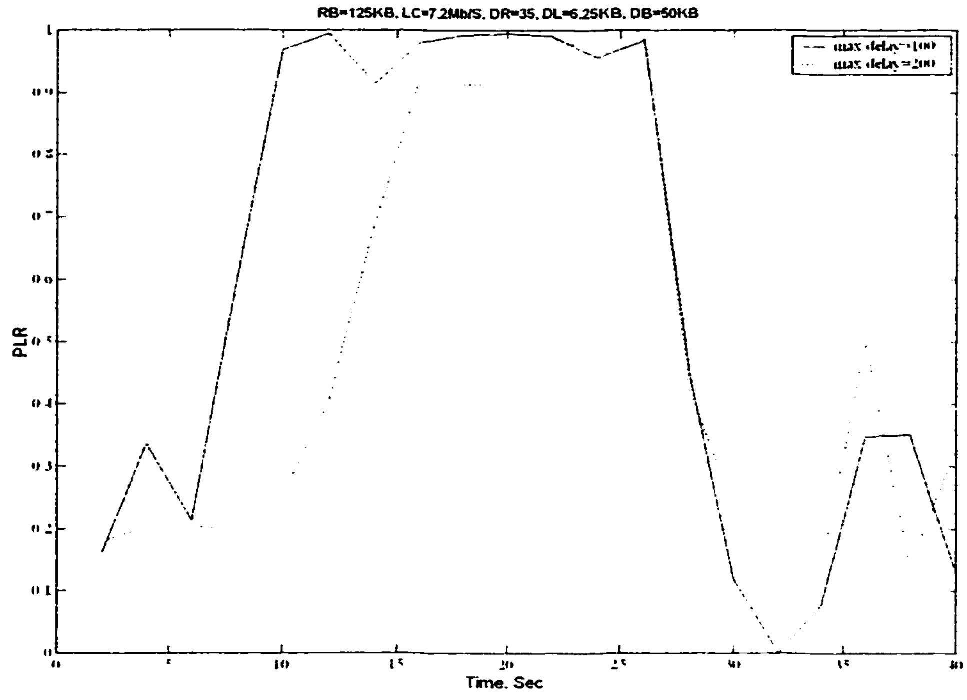


Figure 5-27: Effect of maximum acceptable delay on lost packets

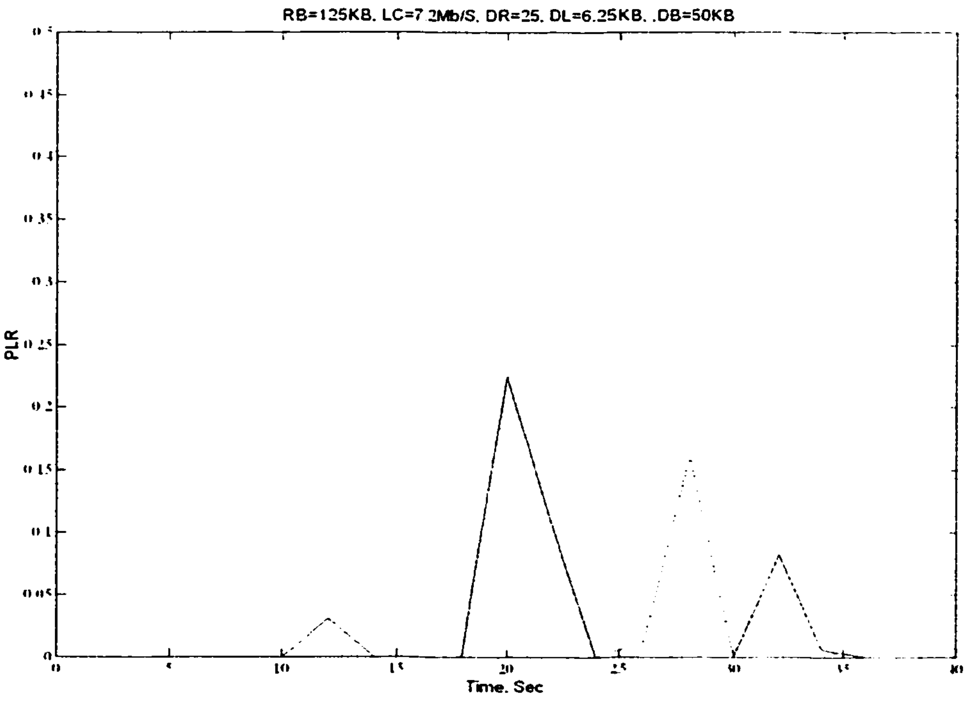


Figure 5-28: Effect of maximum acceptable delay on lost packets

the latency in occupation of the buffers to increase the queuing delay to the maximum acceptable level. It is expected that if we increase the time from 26 seconds, the difference becomes less. For Figure 5-28 although all the packets in both cases are lost due to delay, but number of lost packets decreases by more than 75% with doubling the maximum acceptable delay

	DR=25	DR=35
Max Delay 100	Total lost packs=35433	Total lost packs=308
	Loss due to delay=91%	Loss due to delay=100%
Max Delay 200	Total lost packs=17378	Total lost packs=42
	Loss due to delay=80%	Loss due to delay=100%

Table 7: Effect of maximum acceptable delay on lost packets

5.3.8 Error propagation ratio

It would be interesting to investigate how much in average a lost (or in error) packet is "amplified" or propagated due to intra/inter frame coding nature of MPEG video. As we saw in 3.2, one lost packet carrying MPEG information can have a considerable effect on the video and generally it causes much more than just one packet to be damaged. In this part we calculate how many packets in average are damaged due to loss of one packet. For this purpose we should know how many packets have been sent to the network and how many are lost. Since the simulation gives 13 measurements for PLR during the first 26 seconds, we will be also able to estimate total number of lost packets considering error propagation.

During the first 26 seconds of the simulation, i.e., before adaptation, the frame rate and GoP structure are known (30 f/s and **IBBPBBPBBPBBPBB** respectively). Considering the average frame sizes of section 5.2.1.2 and assuming that the MTU is the average of the two assumed technologies used in the simulation, i.e., average of 1.5k KB of Ethernet frames and 4.5 KB of Token ring frames (i.e., 3KB), a simple calculation shows that for the case that we encapsulate one slice in each RTP/UDP/IP packet, during the first 26th seconds of simulation, totally something around 45000 packets have been generated by the 8 users, i.e., 3500 packets during every two seconds that PLR values are calculated.

Table 8 shows the values measures by the simulation or calculated for the slice case. As we see, in average damage of one packet has damaged 6-7 packets due to dependency of information in and between MPEG video frames in case.

Total packets generated	**Total lost packets**	$3500 \times \sum_{1}^{13} PLR$	**Propagation ratio**
45000	**1812**	**12460**	**6.87**
	1668	**10381**	**6.22**
	1797	**11101**	**6.17**
	1802	**11225**	**6.22**

Table 8: Error propagation ratio

6 Conclusion

6.1 Summery

In this research we investigated streaming of MPEG video over IP networks. Through a simulation, we showed that due to "Best Effort" nature of these networks, the packets carrying MPEG information are exposed to lose or unacceptable delays. We also showed that due to inter/intra frame coding of MPEG video, the effects of packet loss propagates to other frames, based on the information the lost packet carries.

We then described the adaptation techniques that are used to improve the Packet Loss Ratio and net QoS for MPEG video over IP. Especially we investigated the effect of a source driven adaptation technique through a simulation. This adaptation technique works based on a combination of frame dropping and change of MPEG encoder quantizer step size. The simulation showed the effectiveness of the adaptation technique and the improvement of the PLR considerably.

The way that the MPEG video Elementary Stream information is put in RTP packets has been restricted by the standard to an integral (but unspecific) number of slices. Through this study we compared two methods of packing MPEG video frames: Packing a complete frame or packing a complete slice in a packet. Through the simulation we investigated how this alternative packing affects the source driven adaptation scheme. We showed that packing each slice will lead to significantly less Packet Loss Ratio due to slices being the synchronization points in frames. On the other hand we showed that a price should be paid for the packing slices of small frames, which is considerable waste of bandwidth due to more frequently transmission of the packet headers carrying MPEG video ES. We measured the optimum packet size of a frame in case of Ethernet being our layer two technology for QSIF and SIF standards and show how the amount of header information changes with the frame 's size in these two standards

We also measured delay and its variance for an MPEG stream. The delay curve has similarity to PLR because our results showed that the contribution of loss due to unacceptable delay is much more than the other two factors namely congestion and buffer overflow. Variance of delay increases with network's load as shown in 5.3.6.

We showed that increase of maximum acceptable delay does not have a considerable effect on the improvement of the PLR in a loaded network. We also showed that loss of a packet in the case of encapsulating one slice in a packet leads to a total damage of around 6-7 packets.

6.2 Recommendation

In 5.3.3 we showed that putting a complete frame inside an RTP/UDP/IP packet leads to an increase in Packet Loss Ratio as compared to when each slice of a frame is packed in an RTP/UDP/IP packet. On the other hand in 5.3.4 we showed that packing of slices of small sized frames leads to waste of bandwidth due to header information that each packet carries. This helps us to recommend that the optimum way to pack MPEG video Elementary Stream information when our choice is limited to one slice and one complete frame is when we put each slice of I and P frames in one packet, while for B frames an entire frame should be pack inside an RTP/UDP/IP packet.

The simulation results showed that delay was the main cause of loss. This implies that increase of link capacities has more effect in improving network QoS than other factors (increasing buffers' size for example)

6.3 Future work

The work done in this thesis could be expanded in several ways. One interesting issue would be studying that how applying priorities to MPEG video packets can affect the QoS. One may assign different priorities to MPEG frames (I frames having higher priorities than P, and P frames having higher priorities than B frames and all of them more than data). In this way we somehow try to protect the anchor frame from loss/delay in order to reduce error propagation. Logically we expect the PLR improves considerably, so the emphasis here should be on the effect of prioritization on data traffic.

Another interesting way to expand this work is to calculate the optimum packing size for MPEG packets transmitting over IP networks. In this research we limited ourselves to choose between packing one frame or one slice of each frame to see its effects on the adaptation algorithm. The work done here could be expanded to find the optimum packet size (with or without considering an adaptation technique). Optimization could be defined in several ways (PLR, bandwidth, a combination of them...) based on different applications.

MPEG-4 is a rather new standard officially released in 1999. This standard targets low bandwidth applications such as wireless Internet. The way that MPEG-4 compresses information is much more complicated than MPEG-1 and MPEG-2. Transmission of MPEG-4 over IP networks can be another topic for the future research.

Bibliography

[1] j. postel: Internet Protocol, RFC 791

[2] Douglas E. Comer: Internetworking with TCP/IP, Principles, Protocols, and Architectures, 4th edition, Prentice Hall, 2000

[3] Eric D. Siegel: Designing Quality of Service, John Wiley, 1999, pp. 244-5

[4] Larry L. Peterson, Bruce S. Davie: Computer Networks, a system approach, 2nd Edition, Morgan Kaufman, 2000, pp.215-230

[5] Paul Freguson, Geoff Huston: Quality of Service, Delivering QoS on the Internet and in Corporate Networks, John Wiley, 1998, P.58

[6] Douglas E. Comer: Internetworking with TCP/IP, Principles, Protocols, and Architectures, 4th edition, Prentice Hall, 2000

[7] postel: User Datagram Protocol, RFC 768

[8] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP: A Transport Protocol for Real-Time Applications, RFC1899

[9] Eric D. Siegel: Designing Quality of Service, John Wiley, 1999, pp. 2-10

[10] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin: Resource ReSerVation Protocol (RSVP), RFC 2205

[11] Eric D. Siegel: Designing Quality of Service, John Wiley, 1999, pp. 242-250

[12]J.Mitchel, W.Pennebaker, C.Foff, D.LeGall: MPEG Video Compression Standard, Chapman and Hall, 1996

[13] D.Lindberg,R.baker, T.Lookabaugh, J.Gipson, T.Berger: Digital Compression for multimedia, Morgan Kaufman, 1998, pp. 364-390

[14] Official MPEG Homepage, http://mpeg.telecomitalialab.com

[15] R.S.Ramanujan, J.Newhouse: Adaptive streaming of MPEG video over IP networks, IEEE Conf. on Local Computer Networks, Minneapolis, Minnesota, October, 1997

[16] Xue Li et al: Video Multicast Over the Internet, IEEE Network, March/April 1999 PP.46-60

[17] B. Vickers, C. Albuquerque, and T. Suda: Source-Adaptive Multi-Layered Multicast Algorithms for Real-Time Video Distribution. IEEE/ACM Transactions on Networking, December 2000.

[18] S. McCanne, V. Jacobson, and M. Vetterli: Receiver-driven layered multicast in Proc. SIGCOMM'96, ACM, Stanford, CA, Aug. 1996,

pp. 117–130

[19] B. Zheng and M. Atiquzzaman : Two stage frame dropping for scalable video transmission over data networks, IEEE Workshop on High Performance Switching and Routing, May 2001, pp. 43-47

[20] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd: Filters: QoS Support Mechanisms for Multipeer Communications, IEEE Journal on Selected Areas in Communications, Special Issue on Distributed Multimedia Systems and Technology, Vol. 14, No. 7, Sept. 1996, pp. 1245-1262

[21] Michael Hemy M. Hemy, U. Hengartner, P. Steenkiste' T. Gross: MPEG Streams in Best-Effort Networks, Proc. Packet Video'99, April 1999

[22] J.Boyce,R.Gaglianello :Packet loss effects on MPEG video sent over the public Internet, ACM Multimedia 98, pp.181-190

[23] O. Verscheure, P. Frossard, M.Hamdi: User-Oriented QoS Analysis in MPEG-2 Video Delivery, Journal of Real-Time Imaging, October 1999, pp.305-314

[24] D. Hoffman et al: RTP Payload Format for MPEG1/MPEG2 video, RFC 2250

[25] Video Quality Expert Group (VQEG) Home page, http://www-ext.crc.ca/vqeg/frames.html.

[26] I.Dalgic et al: Glitches as a measure of video quality degradation measured by packet loss, in Proc. 7th International Workshop on packet video, Brisbane, Australia, pp. 201-206, 1996

[27] T.Cormen, C.Leiserson, R.Rivest: Introduction to Algorithms, McGraw-Hill, 1997, P.201-2

[28] D.Bertsekas, R.Gallagar: Data Networks, Prentice Hall, 1992, pp.145-150

[29] C. Holmes, D. Denison, M. Hansen, B. Yu, and B. Mallick: Internet Traffic Tends Toward Poisson and Independent as the Load Increases, Bell Labs Internet Traffic research, 2002

[30] Marwan Krunz, Ron Sass, and Herman Hughes: "Statistical Characteristics and Multiplexing of MPEG Streams," in Proceedings of the IEEE INFOCOM '95 Conference, pp. 455-462, April 1995

[31] M. Baldi, Y. Ofek: End-to-End Delay Analysis of Videoconferencing over Packet-Switched Networks, IEEE/ACM Transactions on Networking, August 2000, pp. 479-492

[32] Stephan Thomas: IPng and TCP/IP protocols, John Wiely, 1996, page 94

[33] Eric D. Siegel: Designing Quality of Service, John Wiley, 1999, Page 130

[34] B. Douskalis: IP Telephony, Prentice Hall, 2000, pp.128-131