NOTE TO USERS

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Quanyou ZHOU

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

Department of Electrical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Experimental Study and Performance Analysis of real Time Application Over Differentiated Services Capable Networks

D. Makrakis

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

H. Changcheng

W. Lee

J. Zhao

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Experimental Study and Performance Analysis of Real Time Applications over Differentiated Services Capable Networks

By

Quanyou Zhou

A Thesis submitted to
The Faculty of Graduate Studies
University of Ottawa
in partial fulfillments of the requirements
for the degree of

Master of Applied Science
in Electrical Engineering

Ottawa-Carleton Institute for Electrical Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario, Canada

September 2004

Your file Votre référence
ISBN: 0-494-01662-0
Our file Notre référence
ISBN: 0-494-01662-0

ABSTRACT

Development of technologies for enabling Internet to provide Quality of Service (QoS) are among the most urgent and aggressively pursued research activities in the field of computer communications worldwide. The produced protocols and standards targeting developed for this purpose are better known under the names Integrated and Differentiated Services. Application of these technologies involves use of a variety of new traffic estimation, policing, packet forwarding, resource reservation and connection establishment algorithms, protocols etc. As all these elements interoperate and interact with each other, they generate a variety of effects on the traffic streams and the behavior of the corresponding applications. It is critical that these interactions are discovered and well understood prior to the large-scale deployment of equipment and services. Services involving time and information loss sensitive applications are among the most critically affected by these mechanisms.

The research reported in this dissertation, focuses on the delivery of QoS by Differentiated Services enabled networks to MPEG-2 compressed video and Distributed Interactive Virtual Environments (DIVE) applications. Video based services will be among the most popular high-bandwidth applications running in the future through Internet. DIVE based services are emerging as powerful tools for use in distributed simulation environments and are expected to be used in developing applications for the support of several very promising business sectors; among them e-commerce, tele-learning / tele-training and networked games. MPEG-2 video and DIVE applications are highly sensitive to delays and losses. Our objectives were to: i) assess the ability of Differentiated Services networks to support such applications; ii) determine what algorithms are most suitable; iii) find where certain algorithmic parameters should be placed for best performance; iv) identify potential weaknesses and propose solutions. We performed our research by developing a Linux based Differentiated Services capable network, and by conducting extensive sets of experiments and measurements. As the reader will find out, we succeeded in all the objectives. Performance of the applications and network behavior were successfully measured. "Mechanisms", affecting performance in a negative way, were discovered and solutions were proposed. We also demonstrated the ability to provide advanced networking solutions using off-the-self computing equipment (regular PCs) and open source operating systems (Linux in our case). Such low cost solutions could be very beneficial for supporting the needs of small size business and of tomorrow's "intelligent home".

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. D. Makrakis for all his continuous support, guidance, trust and patience throughout my study and the writing of this thesis. His extensive experience and knowledge in the networking and communications fields are great technical support and precious professional resource keeping me up with the new technologies. My most sincere thanks go to him for giving me the opportunity to improve my knowledge, and for helping me to achieve my goal.

I also would like to thank my colleagues for all the sharing of ideas and thoughts with me and for the feedback they provided me. Special thanks go to Mrs. H. Yu, Mr. T. Yang, and Mr. D. Liu for their useful discussions and help. I am also very grateful to Mr. O. Kabranov for his kind and considerate help throughout all my study.

Finally, I would like to deeply thank my wife for all her support and encouragement through this effort. Her contribution is present in every single page of this work. Without her constant love, support and encouragement this study would have never been completed.

Thank you!

To

Jason and Xiaozhu

Contents

List of Figures

List of Tables

List of Acronyms

AP	Access Point
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
CBQ	Class Based Queuing
CBR	Constant Bit Rate
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DiffServ	Differentiated Services
DIVE	Distributed Interactive Virtual Environment
DSCP	Differentiated Services CodePoint
EF	Expedited Forwarding
AF	Assured Forwarding
BE	Best Effort
FIFO	First In First Out
GOP	Group OF Pictures
IETF	Internet Engineering Task Force
IS	Integrated Services
ISO	International Standards Organization
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MPLS	Multi-Protocol Label Switching
NIC	Network Interface Card
NTSC	National Television Systems Committee
OSI	Open Systems Interconnection
P-FIFO	Priority-FIFO

PQ	Priority Queue
PHB	Per Hop Behavior
PSNR	Peak Signal to Noise Ratio
QoS	Quality of Service
RAM	Random Access Memory
RED	Random Early Dectection
RSVP	Resource ReSerVation Protocol
RTP	Real Time Protocol
SNR	Signal to Noise Ratio
STD	Standard Deviation
TBF	Token Buffer Filter
TC	Traffic Control
TCNG	Traffic Control Next Generation
TCSIM	Traffic Control Simulator
TCP	Transmission Control Protocol
UDP	User datagram Protocol
VBR	Variable Bit Rate
VoIP	Voice over IP
VQ	Virtual Queue
WFQ	Weighted Fair Queuing

1. Introduction

1.1 Quality of Service (QoS) in IP Networks

The explosive growth of communication technologies and deployment of networking infrastructure around the globe have forced us rapidly to enter into the Information Technology (IT) age. Internet based applications are becoming ever present in all our activities, for example, Internet telephony, videoconference and E-commerce. There is no doubt that the Internet will play a more and more important role in our lives.

However, the large amount of applications over networks results in network resource competition. As we know, the current IP (Internet Protocol) network is a packet switched data network and only provides best effort (no discrimination) service to all streams. Therefore it is inevitable for packets to be lost or delayed on the route due to network congestion. This phenomenon has no influence to textual applications such as FTP, WWW etc, but it is intolerable for real-time applications such as Internet telephony and E-commerce because of packet loss and delay sensitivity. For example, in IP telephony application, even a few audio packets deficiencies will incur the loss of important contents. In addition, even we increase the bandwidth to a Gigabit, the traditional best

effort network still cannot guarantee the real-time traffic packet arriving at the destination in the proper order and within the required time.

In order to face such challenges from real-time applications and improve the performance of the Internet, Quality of Service (QoS) is required.

QoS can provide the quality guaranteed service for selected network applications or packets. The goal of QoS is to meet the network application requirements such as bandwidth, jitter and delay and packet loss rate. For example, if video traffic is given priority, the video packets will not experience loss or intolerable delays even under the circumstance of network congestion. [XIA99] [KAR99].

Based on prior QoS development and the probable future need for network QoS architecture, the IETF (Internet Engineering Task Force) has proposed two fundamental internet service models for providing the QoS requirement over the Internet: Integrated Services (IntServ) [BRA94] and Differentiated Services (DiffServ) [BLA98] [NIC98].

The Integrated Services model is based on the Resource reSerVation Protocol (RSVP) [BRA97] [MAN97] [IETF0]. The RSVP protocol can provide QoS for flow based traffic between senders and receivers. The RSVP protocol is responsible for maintaining the application path and reserving the network resources. With RSVP, senders can apply QoS parameters from the network service provider for their applications. The QoS parameters include end-to-end delays, packet loss rates and jitters etc. By sending the QoS request message through the path the packet needs to traverse, RSVP will determine whether each node along the path meets the requirements defined in the QoS request, if not, the sender will get an error message, if so, the path will be set up and the application traffic will traverse along this path to the destination. In order to make the IntServ model work, RSVP needs to maintain and refresh periodically every intermediate node along the path to provide the QoS guarantee for every single flow. The IntServ is kind of soft state service model. Three traffic classes are defined in the IntServ service model: Best effort traffic, controlled load traffic and guaranteed traffic [IETF0]. The limitation for the

IntServ service model is its scalability in large networks because of the heavy traffic load in each node.

As a simpler and more scalable QoS architecture, Differentiated Services (DiffServ) brings a promising light for future network QoS architecture development. By aggregating the traffic classification state, DiffServ dramatically reduces the complexity and state management in every node and it provide an easier way to implement scalability in large networks. Packets are classified and marked by their QoS level using the Differentiated Services Code Point (DSCP) field. In the IETF's definition, the edge router is responsible for the traffic conditioning such as packet classification, policing and shaping, and the core router inside the network is responsible for forwarding the packets, which have the same mark with the same forwarding mechanism. DiffServ, as an important type of QoS architecture, has now been adopted by many router manufacturers. However, the research work on its ability to support real-time applications is still in development.

As an emerging QoS architecture, Multi-protocol Label Switching (MPLS) [ROS99], is also attributed to the efforts of the IETF. MPLS is an advanced mechanism based on the label swap technology. Unlike the traditional IP routing policy, MPLS can control the forwarding path by setting up the data path (LSP, Label Switch Path) through extending RSVP signaling and through path protocol before data transmission begins. MPLS not only provides fast forwarding speed for traffic using its shim header, a 32 bit label, but also provides high QoS guarantees by traffic protection and traffic engineering using the constraint-based routing protocol [AWD99] [XIA00].

As we know, the DiffServ architecture is located in layer 3, MPLS operates in layer2, so the integration of DiffServ and MPLS [HOR00] will provide a wider and stronger QoS platform for future Internet development. In this work, we put our emphasis on DiffServ to explore its ability for supporting real time applications. Obviously, our results will provide a great deal of proof for further integrated DiffServ and MPLS network architecture research.

1.2 Motivation and Previous Work

With the rapid expansion of the Internet, the QoS research work for real-time applications has experienced considerable progress. However, how the new QoS architecture and protocol supports and affects the performance of applications still needs to be explored and verified.

As described in section one, DiffServ architecture can provide a QoS platform for real-time applications over networks from its design objective. However, there is no detailed information about the performance of real-time applications over DiffServ enabled IP networks. Therefore, in order to evaluate and verify DiffServ effectiveness, we need to provide more evidence to demonstrate its robust QoS guarantee for real-time applications, in terms of performance statistics from the application layer and the network layer.

Real-time applications, as defined in [DUN89] and [TOB96], have stringent real time constraints and require predictable quality of service. Research on how to support real-time applications over the Internet with the QoS guarantee has been extensively conducted in recent years. The range of research work is from video and voice network applications to mission-critical data network applications etc.

In [GUP94], [DAL94] and [CSA97], the authors presented the research results of packet video streaming over networks. In [GUP94], by analyzing the results, the authors suggested that a better video compression algorithm and communication protocol should be invented for solving current dilemmas. In addition, the authors stated that protocol data unit is a key component for affecting the overall quality of service. The Simulation results were presented in [DAL94] and [TOB96] for evaluating video, audio and data traffic over 10Base-T networks and Fast Ethernet with modified transport and MAC protocols. Actually their protocol architecture is not suitable for wide deployment. In [EDW94], the simulation results for evaluating the performance of a video-telephony application over the Ethernet were conducted. In order to overcome highly variable

packet delays and higher packet losses caused by the trace traffic model, the author proposed a new delay control scheme, and showed the improvement in simulation results. In [FEA02], the author proposed a packet loss model to recover packet losses for improving the quality of received video. The Mpeg4 video was used for measurement and the simulation results showed that improvement in the packet loss rate could be achieved. However, how real-time video applications over DiffServ enabled networks was not mentioned in these papers.

In [ZDJ89], H.261 encoded video transmission over Ethernet has been investigated. The results of the performed simulations shows that, a large number of real-time video streams with acceptable delay and loss performance can be supported at channel utilization of up to 50-60%.

D.Wu et al [WUD00] investigate the fundamental issues for video communication over packet-switched networks. The end-to-end approach for video transmission has been proposed and the simulation results demonstrate its effectiveness. However, all the results are based on simulation with network path model. No tests are contacted in real network environments.

Research reporting transmission of video and audio through DiffServ capable networks has been reported in several research works (see for example [YUH01] [YAT01] and related references within). However, experimental work to assess the impact of DiffServ networking technologies over audio and video is limited. In [GAL01] the authors investigate the performance of voice over a DiffServ capable network but not video. To the best of our knowledge, no experimental work on video transmission over DiffServ networks based on PSNR video quality evaluation, except ours work, has been reported in the open literature.

In [LAI00], [LAI01], the authors presented the results of the DiffServ evaluation for real-time applications. In these papers, the authors analyzed and evaluated the results they got from the test-bed and proved the DiffServ effectiveness for protecting real time

applications. In [LAI01], the authors investigated the performance of voice over DiffServ capable networks. However, the limitation for their research was that the real-time applications used for measurement are not "real applications", just ordinary udp flows. Therefore in some extend, their results are not totally representative of performance evaluation of real time applications under DiffServ. The detailed research results of Mpeg2 over DiffServ enabled IP networks were presented only in recent years, in prior work [YU012] and [ZH021]. The authors presented a convincing proof for the DiffServ QoS guarantee of Mpeg2 video streaming over networks from the network layer and the application layer.

In [STA02], the QoS evaluation of DiffServ mechanism has conducted in a test-bed environment, the results of the test gave the proof that DiffServ mechanism is effective according to its design purpose. However, the authors put their emphasis on testing EF and AF properties and did not investigate the DiffServ support of real time applications.

G. Garrozzo et al. [GAR02] have investigated the QoS performance evaluation of real time applications in a DiffServ tested scenario. They found that DiffServ could provide high quality level of service when appropriate algorithms and resource sharing are chosen. However, they did not give more comparison results of different algorithms. In addition, the method for evaluating video quality is only limited using Mean Opinion Score (MOS).

Research on DIVE (Distributed Interactive Virtual Environment) applications over networks was rare as well. The DIVE application traffic model was introduced in [GAL99], and the performance of DIVE over Ethernet and ATM networks was also presented. In fact, it provided basis for later research work on DIVE applications. In [QIN99], the author presented the work about DIVE application over DiffServ enabled networks, but the author neglected the fact that performance of DIVE application is often affected by other real time traffic such as video streaming application with the same service level. In [YU011] and [ZH022], the authors addressed above problem and proposed a new traffic control scheme. The results demonstrated the performance

improvement of the DIVE application but still needed to be improved based on the optimizing objective.

1.3 Objective

The main focus of this thesis is the performance analysis of real time multimedia of MPEG2 and DIVE applications over Differentiated Services capable IP networks and exploring the extension of Differentiated Services to wireless environments.

The objective includes four components:

- Performance evaluation of MPEG2 video and DIVE applications over DiffServ capable networks.
- Analysis of real time application performance affected by various network configuration and traffic control techniques.
- Proposing new traffic control schemes or improvements.
- Analysis of the interaction of different traffic classes and their impact on the QoS.

The main tasks we carried out in our research are the following:

- Implementation of a Linux based experimental Differentiated Services testing environment for the performance evaluation of real time application. The work includes the implementation of the DiffServ edge, core routers, the development of traffic control queuing discipline schemes, as well as the development of packet capture and analysis tools.
- Implementation of a MEPG2 based video streaming server and a client, as well Peak Signal to Noise Ratio (PSNR) evaluating system.
- Deployment of the DIVE application for experimental testing
- Experimental performance evaluation and testing the MPEG2 and DIVE applications in a DiffServ environment. We analyzed the testing results and evaluated the performance of the real time application from both network layer (packet loss, delay, and jitter) and application layer (PSNR, MOS) parameters.

- Implementation of a LAN emulator with differentiated services support followed by performance evaluation and testing of the real time application performance under different link conditions.

1.4 Thesis Contribution

The contributions of my work are as follows:

- Tested MPEG-2 based video on demand under a variety of network conditions and examined the best effort and DiffServ architecture. DiffServ was chosen because it is a scalable and cost effective method for offering multicasting services to many users. I examined the performance under different types of traffic and traffic volumes, using a variety of network architectures. In addition, I also linked the application level QoS parameters (frame loss and frame jitter) to network layer parameters (packet loss and packet jitter). This could allow a direct interpretation of the application's quality by simply observing the behavior of the packets at the network layer.
- Having established that the existing DiffServ architecture is not able to support DIVE type of applications, I proposed a novel queue management algorithm, which protects the DIVE applications running under DiffServ. The algorithm makes the deployment of the loss and delay sensitive applications possible. Taking into consideration the importance of DIVE applications to electronic commerce, telemedicine, tele-learning etc., it is quite evident that the deployment of the extended architecture in Internet will be highly beneficial.
- My work has been experimental. It is based on the use of low-cost Linux based PC units, which have been modified to function as routers and video clients. My work proved that such low cost units can be used for developing a networking environment able to support even highly sensitive real time applications such as multimedia and virtual reality.

1.5 Thesis Organization

The remaining chapters of this thesis dissertation are organized as follows. In Chapter 2, we present a brief introduction of the MPEG2 applications and DIVE (Distributed Interactive Virtual Environments). In Chapter 3, we introduce DiffServ and its Linux implementation. Furthermore, the test-bed configuration and software used in our test are introduced in Chapter 4. In Chapter 5, we provide the experimental results and present the performance of the MPEG2 and DIVE applications. Conclusions, along with some suggestions for future work, are given in Chapter 6.

2. Real-Time Multimedia Applications

2.1 Multimedia applications and QoS

Advanced Internet technology and fast workstations brought the revolution of multimedia applications into our daily lives. Traditional multimedia applications such as visual communication and entertainment are migrating to the Internet and computing based systems. No doubt, the existing and emerging applications will have a significant impact on the structure of the Internet, which, in its existing form, is designed for textual data communication. However, due to their real time properties, multimedia applications are sensitive to packet delay and jitter. Thus, when mixing real time multimedia traffic with non-real-time data traffic over the same network, inevitably, the performance of the multimedia application is affected in a negative manner.

In order to support networked multimedia applications, research and development focus on two major directions: multimedia data compression techniques and Internet multimedia network protocol stack development [KAR99].

The data compression of multimedia technology has improved dramatically over the last 20 years. With video compression technologies such as H.261 [LEI94], H.263 [ITU26], MPEG-1 [GAL91], MPEG-2 [ISO95], [HAS97], MPEG-4 [OVE00] [EBR02], we have achieved compression rates that exceed 100 times. This created the possibility for the delivery of high quality digital video through networks. In this work, we used MPEG2 and MPEG4 as video sources for our QoS testing.

The key factor for networked multimedia applications is the network protocol support. Without QoS protocol guarantees, it is impossible to provide extensive and widespread delivery of multimedia applications through the Internet.

Several parameters that influence the performance of multimedia applications are:

Packet Loss: It will severely degrade the quality of the multimedia application and even crash the application. It often happens when the network is under congestion.

Packet delay and Jitter: Delay is the time it takes a packet to transport through the network. It is often referred to as end-to-end delay. It plays a crucial role in video and voice applications. For example, the maximum delay of VoIP is 150ms to 200ms [DUN89] [APO91]. If the packets go over this limit, the voice quality is unacceptable. Jitter is the variation of delay. It means packets are coming in with irregular times. This phenomenon also degrades the performance of the multimedia application. To reduce jitter, besides network solution, a playback buffer can be used at the receiving side.

Significant effort has been made by the IETF to provide QoS architecture to the Internet, thus, enabling it to control packet loss, delay and jitter. One of the types of QoS architecture is Differentiated Services. How DiffServ supports multimedia applications over the Internet is the focus of my research work.

2.2 MPEG Compressed Video

The MPEG video compression standard is aiming to achieve significant reduction of the growing need of generic coding methods for moving images of various applications such as digital storage and communication. Because of the large amount of data produced by digitizing visual information, it is difficult to store as well as deliver the content through networks. In 1988, the Moving Picture Experts Group (MPEG) developed the MPEG-1 standard (ISO/IEC 11172) that supports inter-frame coding [BRA94]. MPEG-1 was designed to produce Video Home System (VHS) quality compressed audio/video at a bit-rate of approximately 1.5 Mbps. The maximum bit-rate can reach as high as 5 Mbps. The increasing need for higher quality and error resilience in video transmission pushed the MPEG committee to introduce MPEG-2 (ISO/IEC 13818, [ISO95]) in 1995, targeting 2M to 8Mbps. An important feature for the MPEG2 is the support of full screen interlaced (TV) or progressive video (computer monitor) at 25-30 pictures/sec. The MPEG2 coding algorithm can dramatically reduce the required bandwidth for image transmission. Today, MPEG-2 is the most widely accepted standard in digital broadcast TV and High Definition Television (HDTV). The MPEG4 video compression standard was put forward by the MPEG group in 1999. It provides a highly efficient video compression method for making compressed video rates from extremely low data rate to bit rates and quality levels beyond the level achieved by MPEG2. It is suitable for use in the wireless environment due to its low bandwidth requirements.

2.2.1 MPEG2

The ordinary MPEG standard supports three source image formats: 4:2:0 format, 4:2:2 format and 4:4:4 format. Each format means different bit rates. For example, the uncompressed bit rate of 720X576, 25 frames/sec for 4:2:2 is 166Mbit/s and for 4:2:0 is 124 Mbit/s. With MPEG 2, the bit rate for 4:2:0 can be decreased to the range of 3-15 Mbit/s. From this example we can see the impressive power of the MPEG compression algorithm. The techniques used for compressing MPEG video data are based on motion estimation and compensation. In the MPEG standard, one full picture may consist of m-th video frames (normally m=12). Discrete Cosine Transform (DCT) [RAO90] is used to

reduce the spatial redundancy in the individual video frames. Motion Compensation is used in order to reduce the temporal redundancy between successive video frames. To ensure that the DCT and motion estimation algorithms are running well and to ensure that every encoder and decoder is working at same pace, the MPEG standard defines three types of pictures for calculating the estimation:

- *Intra-pictures (I-frames):* Self-contained JPEG-encoded still pictures. These pictures are encoded only in respect to themselves and can be used as starting points for decoding [ISO95]. Because of their spatially compressed property, they tend to generate a considerable part of the total data, which the compressor produces.
- *Predictive pictures (P-frames):* These pictures are encoded using motion-compensated prediction from a past I-frame or P-frame. They usually produce less traffic smaller than the I-frames because of temporal reduction.
- *Bi-directional pictures (B-frames):* These are pictures encoded using motion-compensated predictions from a past and/or future I-frame or P-frame.

With the definition of the above pictures, the video bit stream can be divided into a series of Groups of Pictures (GOPs). Each GOP normally contains one or more I-frames, P-frames, and B-frames. A common GOP structure can be described by two parameters: N and M. N is the number of pictures in a GOP. M is the frame number between pictures. Figure 2.1 shows the common GOP structure with the frame order I-B-B-P-B-B-P-B-B-P-B-B (N=12, M=3). Actually, the frame storage order is different from the frame display order. For example, while the display order is as follows:

$$B_1, B_2, I_3, B_4, B_5, P_6, B_7, B_8, P_9, B_{10}, B_{11}, P_{12}$$

The corresponding storage bit stream order is as follows:

$$I_3, B_1, B_2, P_6, B_4, B_5, P_9, B_7, B_8, P_{12}, B_{10}, B_{11}$$

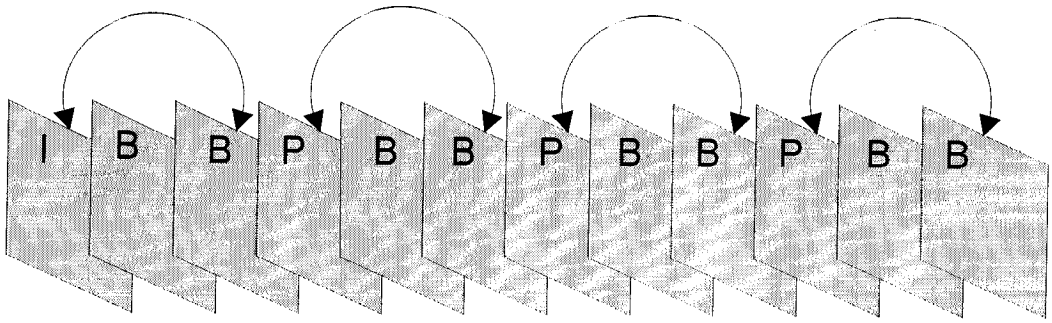


Figure 2.1: MPEG2 Group of Pictures

With the MPEG scalable extension, MPEG2 also supports multi-layer video applications. Scalable extensions of the MPEG2 standard [ISO95] include: Spatial scalable extension, SNR scalable extension, Temporal scalable extension, and Data partitioning extension. In scalable video coding, based on the network conditions or other limitations, the video decoder will decode and display the corresponding layered video data from the coded video. In the case of basic scalability, two layers of video are defined from a given video coded stream. One is the lower (or basic) layer; another one is the enhancement layer.

Typically, the data produced by an I frame is three times larger than that of a P frame and six times larger than the data produced by B frame. This produces variability of the traffic stream volume with time and consequent burstiness. This kind of traffic is hazardous to the network's performance since it stresses the network's resources. We call this kind of traffic VBR (Variable Bit Rate). We can use an adaptive Q-factor compression algorithm (Adaptive Quantization Parameter) to compress the original video as CBR (Constant Bit Rate), however, this will introduce degradation to the quality of the picture.

2.2.2 MPEG4

The MPEG4 video compression standard was put forward by the MPEG group in 1999. In contrast to MPEG2, MPEG4 is aimed at video applications used in low bandwidth or low storage capacity environments designs using model-based image coding schemes and object-based representation. MPEG4 provides highly efficient video compression. It is able to produce extremely low data rates and quality levels beyond those of MPEG2. Its high compression rate and effective error resilient coding (15% of the size of a standard

file, even at 640x480 resolutions) made it become the universal language among movie, broadcasting and multimedia applications.

The MPEG4 divides a video frame into one or several objects, which form a tree-like structure. MPEG4 components are:

- Video and Audio objects
- Object description elements with BIFS (Binary Format For Scenes), which describes the relation between objects.
- Synthetic speech
- Face movement parameters
- Synthetic music
- Text with property parameters

Figure 2.2 shows a MPEG4 scene with multiple objects relation.
MPEG4 video compression rate can reach extremely high values, thus producing very low data rate streams (5-64kbps, suitable for wireless environment). The maximum limit for a video stream in MPEG4 is 4 Mbps.

The techniques used for compressing MPEG4 video data are similar to those used for MPEG2: motion estimation and compensation and Discrete Cosine Transform (DCT). DCT reduces the spatial redundancy in the individual video frames. Motion Compensation reduces the temporal redundancy between successive video frames.

As in MPEG2, MPEG4 also defines three types of picture: *Intra-pictures (I-frames), Predictive pictures (P-frames), Bi-directional pictures (B-frames).*

In MPEG4, the video bit stream can be divided into series of Group of Pictures (GOPs). Each GOP normally contains one or more I-frames, P-frames, and B-frames. Figure 2.3 shows the MPEG4 GOP structure.

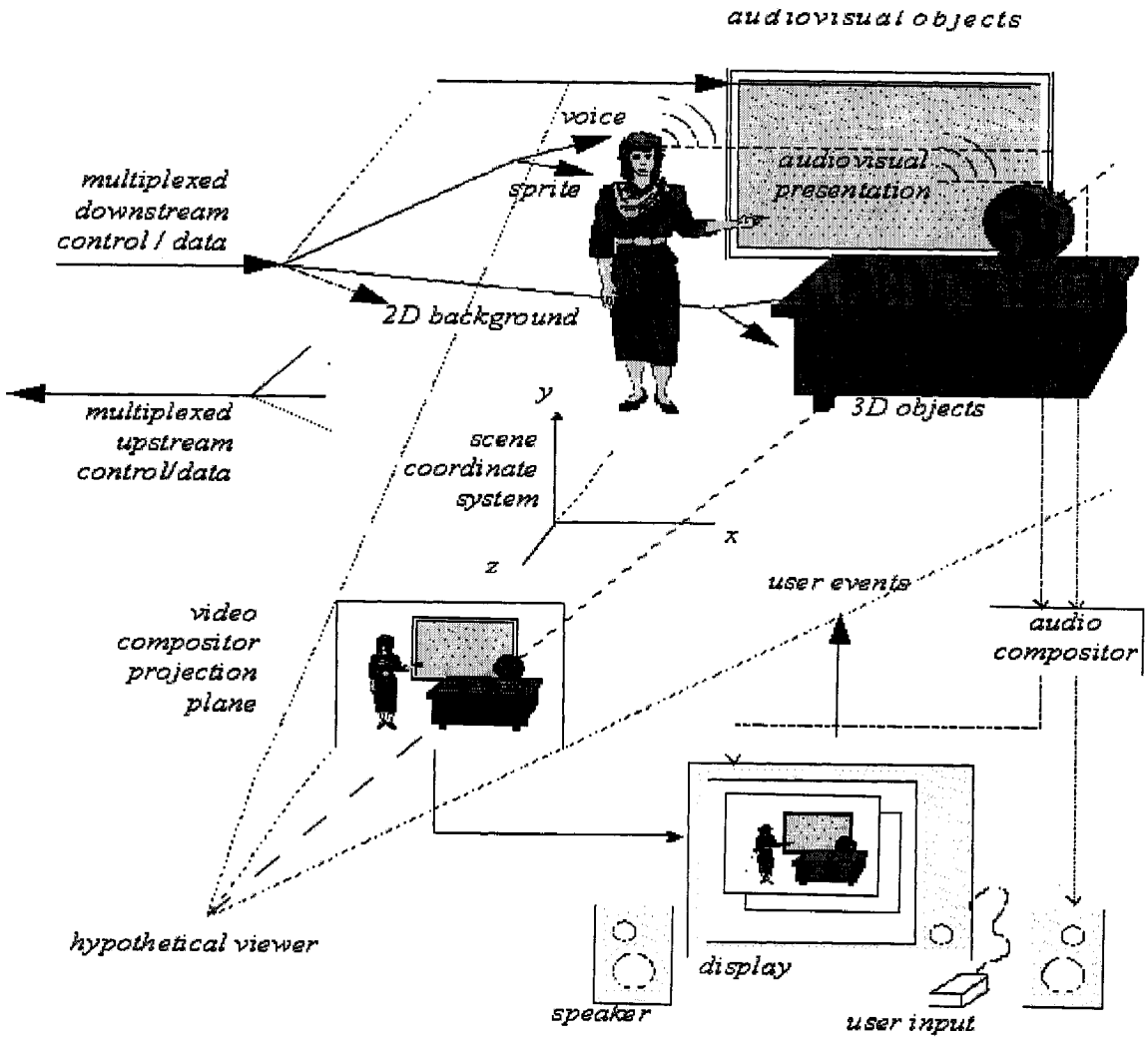


Figure 2.2: An Example of MPE G4 scene ISO/IEC 1998

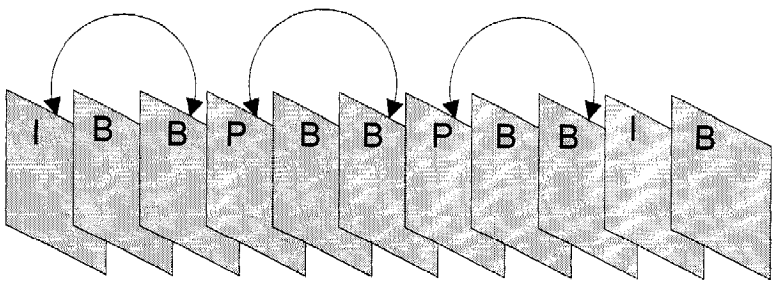


Figure 2.3: GOP for *MPEG4*

As an emerging digital multimedia standard with advanced object structure, MPEG-4 is aimed at providing an integrated object-oriented representation of multimedia content for network multimedia applications such as Video over Internet, multimedia broadcasting,

virtual reality application and video/voice over wireless networks. For authors, network service providers and end users, MPEG4 provides desirable performance for their requirements with great reusability, flexible content production, transparent signaling protocol messaging and low resource consumption ability.

2.2.3 MPEG Video Quality

Generally, video quality is related to personal opinion, so the user's satisfaction is an important criterion for evaluating the quality of video transmission. However, objective indexes are needed as well. In order to evaluate video quality when the streams pass through the network, we use in our work two objective video quality measure indexes.

PSNR (Peak-Signal-to-Noise Ratio): It measures the image quality with an objective way, pixel by pixel. It is defined as the ratio of the maximum value of an image to the RMSE (Root Mean Square Error) [ANS96] [VID00]. The formula is as follows:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j}^{N}\left(f_{ij} - F_{ij}\right)^2 \tag{2-1}$$

$$RMSE = \sqrt{MSE} \tag{2-2}$$

$$PSNR = 20 \cdot \log\left(\frac{b}{RMSE}\right) \tag{2-3}$$

f is the original image with $M \times N$ pixels (transmitted image) and F is the image reconstructed from the receiving stream (received image), i, j represents the horizontal and vertical locations of a pixel, $f_{i,j}$, $F_{i,j}$ represent the value of the pixel at location (i, j), $M \times N$ identifies the dimension of the image, and b is the peak value for a pixel (normally, $b = 255$). The formula shows that PSNR is related to image difference, so PSNR can be a QoS index for analyzing the quality of service obtained by a network

measured at the application layer over the network. In our work, PSNR is used for evaluating MPEG2 video quality.

MOS (Mean Opinion Score) is the result of a perception based subjective evaluation described in ITU-R BT Rec.500 [ITU02]. The number of MOS levels is based on the user sensation about video quality perception. It is divided into 5 levels (5-imperciptible, 4-perceptible, but not annoying, 3-slightly annoying, 2-annoying, 1-very annoying). MOS can be used at application level to evaluate the perceived user quality. As one of most powerful measurements, MOS metric can provide a convenient method for evaluating video quality anywhere under a real time environment, especially in the streaming video application. In this work, MOS is used for evaluating the MPEG4 video application by obtaining how good MPEG video looks to the audience.

2.3 Distributed Interactive Virtual Environments (DIVE)

Distributed Interactive Virtual Environment (DIVE) [DIVE0] [CAD93] is an internet-based multi-user Virtual Reality system where participants share a 3D virtual space and work, play and interact with other users. In DIVE applications, a simulated world runs not on one computer system, but on several, with the networked participants interacting through a network (e.g. Internet). People using this technology are able to interact in real time, sharing the same virtual world. Each of the machines participating in the simulation of the virtual word is called a "host". On each host there is a number of "entities" (things in the virtual environment) that communicate their changing state by sending "update messages". The specific entity that corresponds to a participant's virtual body is called "avatar". Avatars are either included within the virtual world during initialization or dynamically created at a later time. Figure 2.4 shows the DIVE operation over Internet.

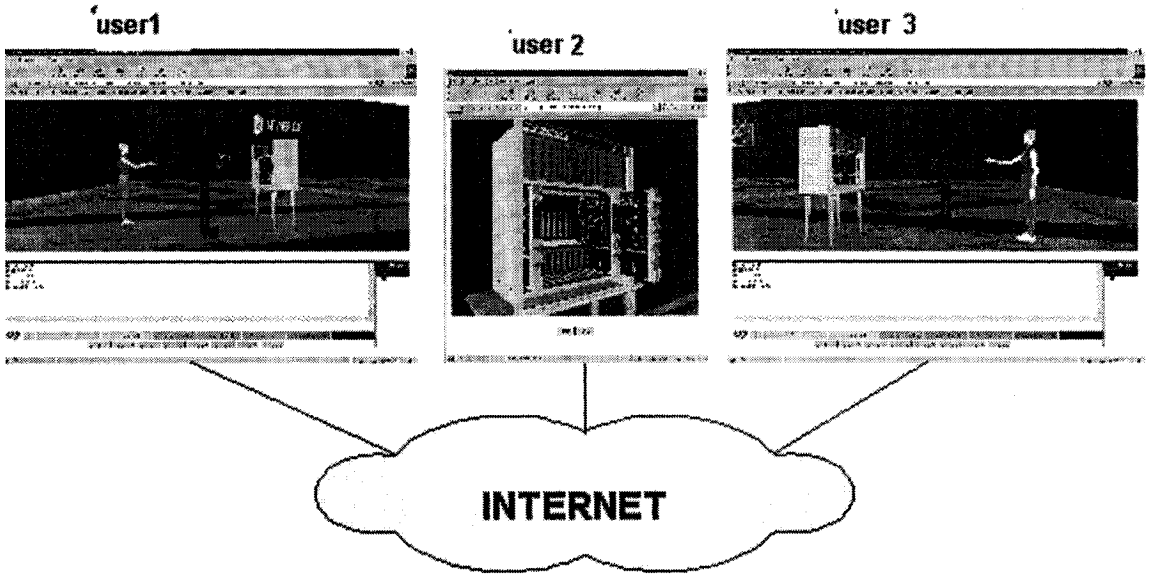


Figure 2.4: DIVE over Internet [YUH012]

The following are some of the key features of DIVE applications:

- *They are sensitive to packet delays.* Any action issued by any participant in the DIVE (transported through packets) must reach the other participants within 100 ms. [CADIS]
- *They should be capable of scaling to large number of participants.* The number of participants should be unlimited to allow everybody to enter the virtual world.
- *Messages are transported through short packets (few tens of bytes) and frequently.* This characteristic differs from the behaviour of other real-time applications such as audio and video.
- *They demonstrate high level of dynamicity in group structure and topology.* Participants should be able to join and leave the session dynamically.

A large-scale DIVE design proves itself to be challenging and requires innovative solutions at various layers of the system architecture. The most significant aspects among these solutions are the changes, which need to be made at the network layer that is the fundamental element in constructing the networked DIVE applications. DIVE's network

communication architecture is essential, because it is concerned with the multi-user technology abilities of the system in terms of sharing objects and passing messages between those objects. If the networking needs of the DIVE application are not thought over carefully, it will suffer serious degradations. For deploying large-scale DIVE applications successfully, we need to provide robust network architecture, capable of satisfying the strict delay and packet loss requirements of these applications.

2.4 Summary

As large amount of multimedia applications migrate to Internet, the existing Internet network architecture is inadequate to let the global network fulfill its growing obligations in the future. The new Internet needs powerful, yet scalable and simple QoS enabling technologies. In this chapter, real-time applications and their QoS requirement were described and discussed (MPEG2, MEPG4, and DIVE). We will see in the following chapters if and how the prominent QoS supporting Internet technology, Differentiated Services can service successfully these applications.

3. Differentiated Service & Linux Implementation

3.1 Differentiated Services Architecture

The powerful combination of multimedia applications and modern internetworking technologies has produced new advanced Internet services (IP-telephony, video-conferencing, video-on-demand, mission critical applications etc.). The quality of service needs of these applications is quite different from those of textual data. To meet the needs of real-time applications forming multimedia services, sophisticated traffic and resource management mechanisms are needed. To serve these needs, various QoS approaches have been developed. A very popular approach of QoS support in Internet is Differentiated Services (DiffServ RFC 2475) [DIFF00], introduced by the Internet Engineering Task Force (IETF). It can offer advanced traffic management and congestion control capabilities over IP based networks with great scalability to large user population.

3.1.1 DiffServ Definition

Differentiated Services is introduced by IETF as a simple and scalable approach to provide QoS over IP networks. The DiffServ architecture is based on the idea of categorizing IP traffic flows into aggregates and prioritizing the traffic aggregates instead of traffic flows individually. This eliminates the need of maintaining per flow state and signaling at every hop.

IP packets are marked with different priorities for getting specific forwarding treatment, per hop behavior, at the boundary router [BLA98] [HEI99]. A bit-pattern, DiffServ Codepoint (DSCP) in each packet header, inside the Type of Service (ToS) byte is used for marking different priorities in every packet. Figure 3.1 shows that the 8-bit ToS byte is redefined as Differentiated Services (DS) field that is composed by 6 bits DSCP and 2 bits that are currently unused. By the priority DSCP indicates in every packet, the DiffServ core router will map the packet's DSCP to a per-hop behavior (PHB) and arrange corresponding scheduling services for the aggregate flow.

Figure 3.1 Structure of for Differentiated Service Code Point

3.1.2 DiffServ Domain Structure

The DiffServ domain is composed of a contiguous set of nodes (routers) that provide Differentiated Services. Three different types of DiffServ router are defined in DS domain:

Ingress Router (Boundary Router): It is located at the entry point of DS domain and connects the DS domain with other DS domains or non-DS domains. It is responsible for classifying, marking the incoming traffic, performing more complex traffic shaping and policing.

Egress Router (Boundary Router): It is located at the exit point of the DS domain and connects the DS domain with other DS domains or non-DS

domains. It is responsible for traffic conditioning to ensure the outgoing traffic meets the agreement between ISPs or users.

Interior Router (Core Router): It is responsible for packet scheduling and forwarding. It should be able to provide enough bandwidth to different behavior aggregate traffic classes but not exceed the limit defined in the profile.

It is obvious that different DiffServ router has different functions to perform in order to provide the network with the ability of DiffServ services. The main functions include:

- Packet classification, Marking and Conditioning.
- Packet forwarding, Per-Hop-Behavior (PHB).
- Service Level Agreements (SLAs)

DiffServ Components:

Packet classification, Marking and Conditioning component

- **Classifier:** It is classifying the incoming packet into different classes. It is a key component to the providing of differentiated services. The classification may be based on the DSCP only (behavior aggregate classifier) or on multi-fields within the IP-header, such as IP addresses, port numbers and protocols types (multi-field-classifier).
- **Meter:** It is measuring the incoming traffic and compares the measured traffic against a traffic profile, which specifies the user's application's network traffic parameters negotiated between the user and the network service provider, in order to determine whether a given packet stream class is within the specified negotiated service profile or out of the profile.
- **Marker:** It is changing the incoming packet's DSCP within the IP header according to defined rules within the traffic profile.
- **Shaper:** It is policing the incoming traffic for its compliance to the traffic profile by delaying packets. The shaper is also important for controlling the traffic bursts if necessary.

- **Dropper:** It is dropping the packets, based on defined rules within the traffic profile, when the incoming traffic rate is over the rate limit defined in the traffic profile.
- **Traffic profile**: It is specifying the temporal properties of a traffic stream selected by a classifier and provides rules for determining whether a particular packet is in-profile or out-of-profile.

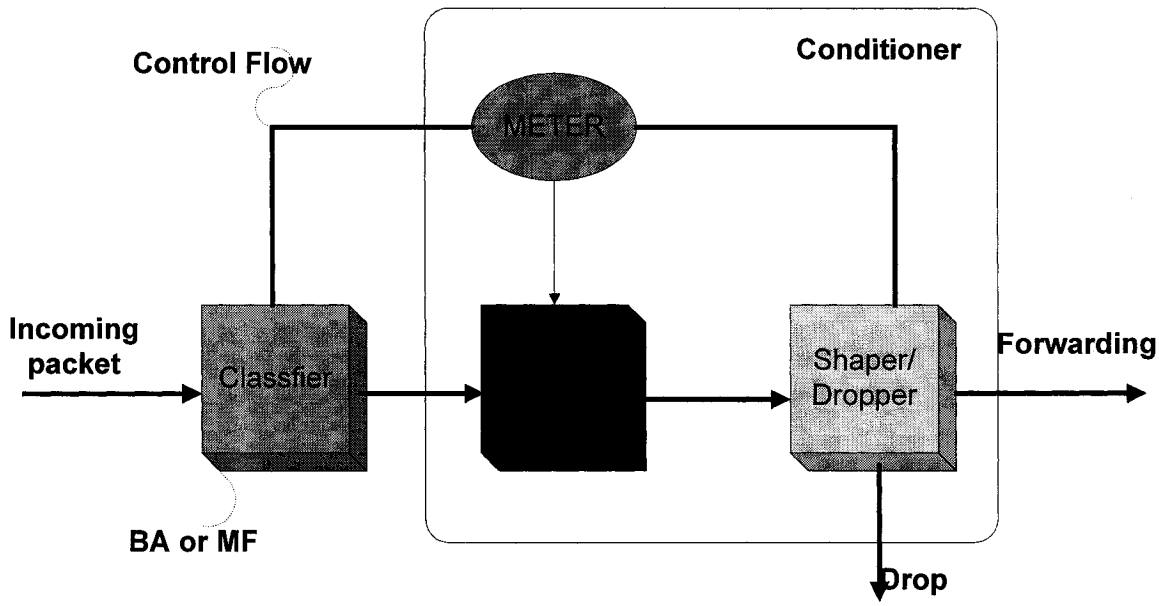


Figure 3.2 Relation between different components.

3.1.3 Packet Forwarding Component

In the DiffServ architecture, Per-Hop-Behavior (PHB) is a key component of the packet forwarding mechanism applied in a DiffServ-enabled router. The definition of Per-Hop-Behavior, as stated in [RFC36], is "a description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate". A specific PHB corresponds to the DSCP marked on the received packet in the DS domain and determines the service the packet will receive.

PHBs are implemented in networks nodes by means of specific packet scheduling mechanism, buffer management and so on. In fact, for supporting PHBs, Network nodes often use different queuing discipline such as priority Queuing (PQ), Class Based Queuing (CBQ), Random Early Detection (RED) and WFQ (Weighted Fair Queuing). Currently, two PHBs have been proposed by the IETF DiffServ working group in RFC2597 and RFC2598. They are: Expedited Forwarding PHB Group (EF) and Assured Forwarding PHB Group.

Expedited Forwarding (EF) PHB

Expedited Forwarding (EF) PHB is also known as premium service. This service is designed to provide low loss, low delay and low delay jitter under a guaranteed bandwidth. It appears to the endpoints like a "virtual leased line" or a point-to-point connection. To achieve the above performance, the queue for expedited forwarding traffic must be very short or almost empty. This means that the fluctuation of this traffic should be very close to within what network has assumed and allocated resources. In addition, because the network is treating EF with high priority, it is important to conform to the SLAs in order not to make resources away from other classes, leading them to starvation. Thus admission control and policing are needed for the EF classes. The ISP guarantees the availability of negotiated bandwidth by reserving a portion of the total network capacity and discards the excess traffic at the boundary nodes. The DiffServ core router provides EF traffic with the highest priority among all traffic classes to ensure its forwarding speed. Recommended DSCP value for EF PHB is 0x2b (101110). In RFC2598, three types queue-scheduling mechanism are recommended: priority queuing, Round Robin and CBQ.

Assured Forwarding (AF) PHB

Assured Forwarding (AF) PHB provides a delivery service for IP packets with four separated forwarded AF classes. Within each AF class, three different treatments can be applied to the packets by using different drop precedence. This means that the traffic can be transmitted through the network with a high probability based on the negotiated

bandwidth in the profile. The exceeding traffic will be marked and forwarded with high priority over the network than the best-effort traffic, but the dropping probability of AF traffic is much lower than best effort traffic. Four different Assured Forwarding classes have been defined with different resource allocation rate and different forwarding assurance level. Three dropping precedence values (low, medium, high) are defined in each class and the packets with higher drop precedence value will be dropped to protect higher priority traffic when congestion happens. The recommended queue management for AF classes is Random Early Detection (RED) (RFC2598).

The AF PHB can be used to implement "the Olympic service", which consists of three service classes: gold, silver and bronze. They can be mapped to the AF classes1, 2, and 3. Table 3.1 lists the recommended AF PHB values and corresponding service level.

	Class 1 (Gold)	*Class 2 (Silver)*	*Class 3 (Bronze)*	*Class 4*
Low Drop Precedence	*001010*	*010010*	*011010*	*100010*
Medium Drop Precedence	*001100*	*010100*	*011100*	*100100*
High Drop Precedence	*001110*	*010110*	*011110*	*100110*

Table 3.1: AF PHB values verus Class Service Level

SLAs and TCAs

Although PHB can determine the traffic aggregate, the rules how to treat the incoming traffic and how to determine the service type still depend on parameters defined in the Service Level Agreement (SLA). The Service Level Agreement (SLA) is a service contract that specifies the forwarding service the customer should obtain from the Internet Service Provider (ISP). It specifies the service classes supported and the amount of traffic allowed in each class. This process administered by SLA is called Service allocation. Customers can specify the service category such as Premium Service or Assured Service and define performance guarantees such as delay loss etc for their application in SLA. SLA can be static or dynamic depending on the designing purpose

(RFC2475). The Traffic Conditioning Agreement includes traffic conditioning rules such as classifier rules and traffic conditioning profile.

3.2 Common Queuing Disciplines

As we mentioned above, in order to implement DiffServ PHB inside routers, several queuing disciplines must be deployed inside the DiffServ nodes. Queuing discipline is embedded in each network interface. Its main function is to control how packets are enqueued on that particular interface. It applies certain policies to the packet passing this network interface, such as classified, be queued or dropped etc. First the Queuing discipline uses filters to identify the different types of traffic streams such as IP addresses or DSCP values, and classifies them into different classes, PHBs. The queuing discipline will process different classes differently, with different queue priority, different queue serving algorithms or rate etc. these parameters are important for allocating the network resource among different classes.

It is obvious that queuing discipline is directly linked to the DiffServ PHB implementation and the QoS provided by the DiffServ. In the following section, we will introduce five types of queuing discipline that are most widely used in DiffServ architecture.

3.2.1 Class Based Queuing (CBQ)

The Class Based Queuing algorithm (CBQ) [FL951] [FLO98] was proposed by Floyd and Jacobson for link sharing.. CBQ can isolate real time and best effort traffic by splitting the traffic into different traffic classes and handle the classified traffic according to its service requirement. As a hierarchical link-sharing model, CBQ determines the controlled distribution of excess bandwidth. An example describing how CBQ operates follows:

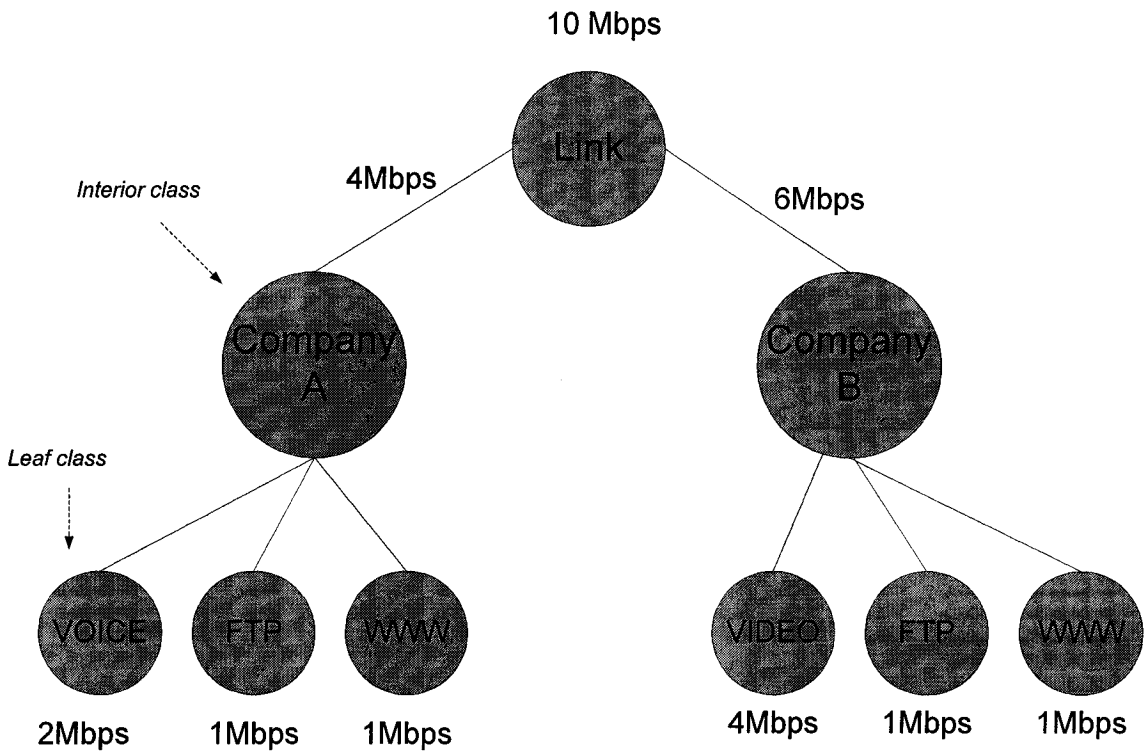


Figure 3.3 A hierarchical link-sharing structure

In Figure 3.3, 10Mbps link capacity is shard by two companies. Company A gets 4Mbps and company B gets 6Mbps when they are in full load. Within the company, the allocated bandwidth is distributed to the different traffic classes such as video, ftp, www etc.

CBQ scheduling uses two types of schedulers:

- A general scheduler, which is used in absence of congestion.
- A link-sharing scheduler, which is used for leaf classes with rate-limiting classes during congestion periods.

However, most schedulers are implemented with Weighted Round Robin or Deficit Round Robin (WRR or DRR).

In CBQ, based on the different characteristics, the traffic can be classified as :

- *regulated*: packets of the class are scheduled by the link-sharing scheduler.

- *unregulated*: packets of the class are scheduled by the general scheduler.
- *overlimit*: the class has recently used more than a specified fraction of its allocated link-sharing bandwidth.
- *underlimit*: the class has received less than its bandwidth-share.
- *at limit*: the class is neither overlimit nor underlimit.
- *unsatisfied*: A leaf class that is *under-limit* and has a persistent backlog
- *satisfied*: A class which is not unsatisfied.
- *bounded*: A class that is not allowed to borrow from ancestor classes.
- *isolated*: A class which does not allow non-descendant classes to borrow its unused bandwidth and also does not borrow from other classes.

The link-sharing goals defined by Floyd and Jacobsen are as follows:

1. Each interior or leaf class should receive roughly its allocated link-sharing bandwidth over appropriate time intervals.

2. If some leaf or interior classes are not using their allocated bandwidth, the distribution of any excess bandwidth among other classes should not be arbitrary. It should be distributed under controlled guideline.

In order to follow the defined goals, certain link-sharing guidelines are introduced for implementing the desired behavior [FL951]:

1. A class can continue unregulated if one of the following conditions holds:
   - The class is not over-limit, or
   - The class has an under-limit ancestor whose level is at most Top-Level. Otherwise, the link-sharing scheduler will regulate the class.
2. A regulated class will continue to be regulated until one of the following conditions hold:
   - The class is under-limit, or
   - The class has an under-limit ancestor whose level is at most Top-Level. Otherwise, the link-sharing scheduler will regulate the class.

CBQ is a powerful scheduling algorithm, especially for real time traffic. It can support different classes with different priorities, which actually can control the delay experienced by the sensitive traffic [FL951] [FL952].

3.2.2 PQ (Priority Queuing)

The Priority Queuing discipline is composed of multiple queues (often FIFO) with different priorities. Queues are serviced by the strict priority order from high to low. Packets from high priority queue are first transmitted to the link. Only when all higher priority queues are empty, a certain queue can receive service. Packets are processed in each queue in FIFO order. This mechanism is important for real time, delay sensitive traffic, however, it can result in starvation for lower priority queue if higher priority queues have large amount of data to be transmitted.

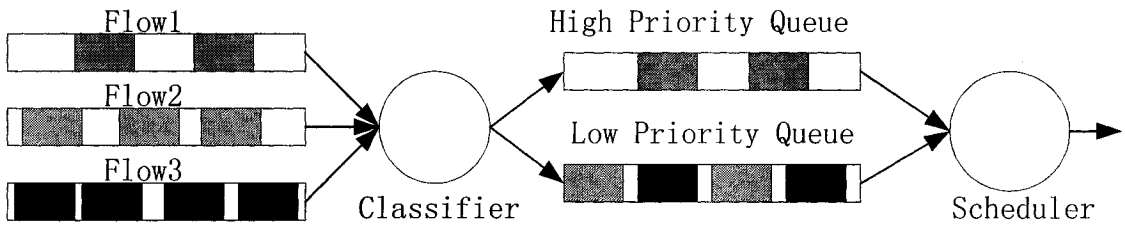


Figure 3.4 Priority Queue

PQ can provide the QoS guarantee for given traffic by assigning it as high priority class. The starvation problem can be solved by limiting the amount of high priority traffic transmitted to the network.

3.2.3 First In First Out (FIFO)

FIFO queuing discipline, as its name indicates, processes packets in the same order they arrive at the queue. It is the simplest queuing discipline in traffic control and is usually used together with other queuing disciplines. In FIFO, all packets are treated equally and placed in one queue until the buffer space has been used; new arrivals

finding the buffer full are dropped. As the packet processing is based only on the order of packet arrival, FIFO does not discriminate packets coming from different traffic sources, nor it provides differential treatment. [SEC01]

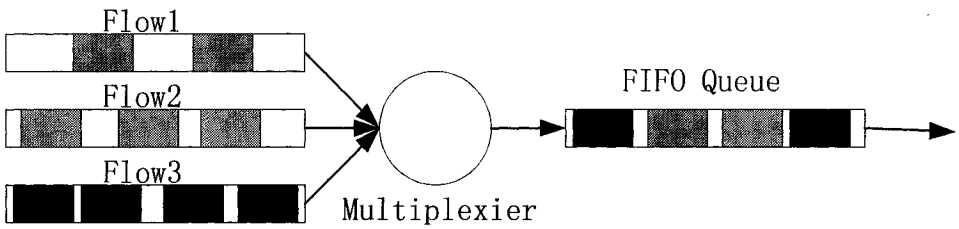


Figure 3.5 First In First Out (FIFO) Queue

3.2.4 RED/GRED

The RED queuing discipline [FLO93] is a congestion avoidance and congestion control algorithm. It is designed for TCP/IP with following objectives:

- Detect the beginning of congestion
- Prevent sustained congestion

RED reacts to congestion either by dropping packets arriving at the router or by downgrading their classification (by changing the value of the appropriate bit in the packet header). The algorithm detects congestion by computing the average queue size, which has a minimum and a maximum threshold. The average queue size can be determined using a variety of algorithms. When the average queue size is below the minimum threshold, all arriving packets are accepted into the queue. As the average queue size exceeds the minimum threshold level, packets are marked or discarded with increasing probability until the average queue size reaches its maximum threshold. All incoming packets are discarded after average queue size exceeds its maximum threshold. Each packet drop serves the purpose of indirectly notifying the source host's transport layer to reduce its sending rate. It is evident that the "design" (configuring) parameters of the RED algorithm are: minimum threshold, maximum threshold, and discard probability. It should be noticed that the gradual change in the value of dropping probability as the average queue size varies between minimum threshold and maximum threshold does not have to be linear.

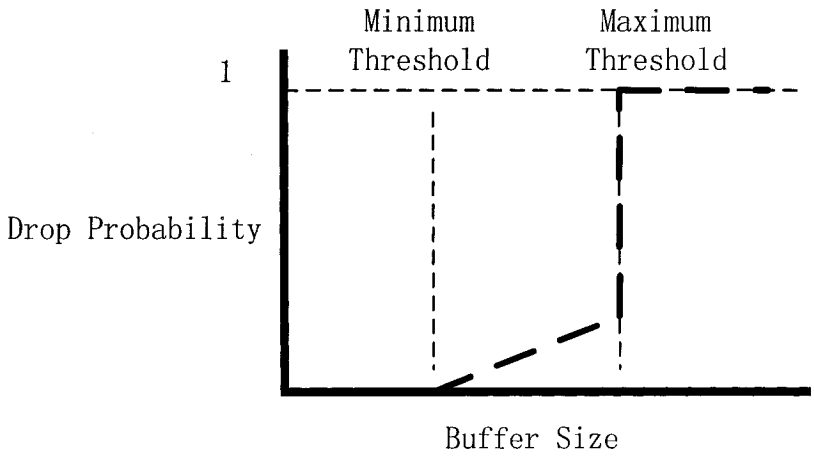


Figure 3.6 Random Early Detection behaviors

WRED (Weighted RED) is an extending RED algorithm. WRED uses a weighted probability value to mark or discard packets. The weights are QoS related parameters. Co-worked with classification/marking /policing mechanism which marked the packets with different drop priorities, WRED can manage the buffer by dropping packets with corresponding probabilities. It is suitable for Assured Forwarding class traffic.

3.3 Differentiated Services Linux Implementation

As mentioned earlier, Differentiated Services intend to provide simple, scalable service discrimination in Internet by reducing the message complexity and state information maintenance. As a QoS-enabled open source operating system, Linux is an ideal platform for integrating the Differentiated services into its kernel, thus providing more powerful QoS control. Detailed information regarding implementation of DiffServ in the Linux environment, description of the Linux network basics and discussion of traffic control can be found in [WER00] [RAD99] [ADE02].

3.3.1 Linux Network fundamentals

Linux is a 32-bit multitasking, multimedia operating system with open source code. Linux is a clone of the Unix operating system that runs on desktop computers. It

provides a large variety of applications software, from X windows to GNU C/C++ complier and TCP/IP.

A complete Unix programming environment also is ported to Linux, including standard libraries, programming tools, compilers, and debuggers. The Linux network implementation provides us with an IP based platform. Figure 3.7 illustrate the path followed by packets through the system.

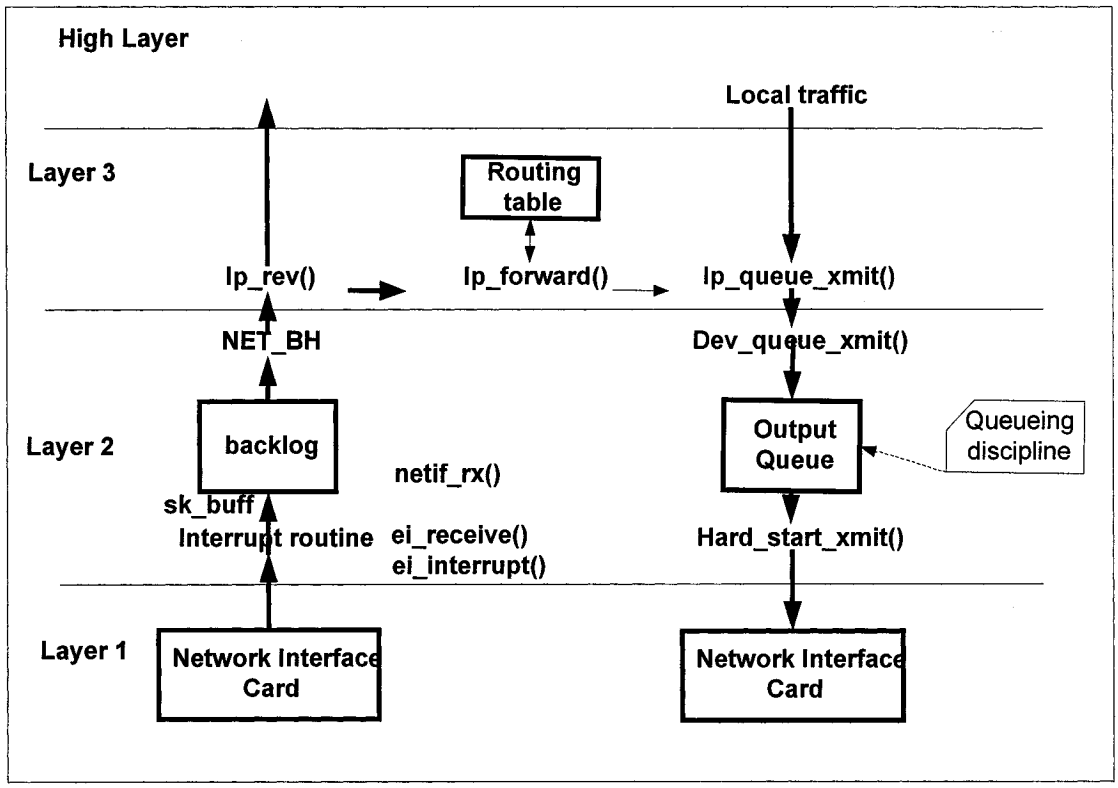


Figure 3.7 Packet path inside Linux kernel

First, an incoming packet will trigger a network card hardware interrupt. The interrupt service routine (*ei_interrupt()*) is invoked and the packet receiving routine (*ei_receive()*) is called for handling the incoming packet by copying the packet from network card buffer to an internal socket buffer (*sk_buff*). After that, a procedure *netif_rx()* is invoked and queues the packet into a central queue (backlog) that holds all the incoming packets from any network interface of the system. After the first stage, the NET_BH routine is checking the output queue status in the network adapter. If there are packets in the backlog queue, the NET_BH handles this packet with the appropriate

protocol. (Such as IP). The function *ip_rev()* is responsible for processing the packet related to the IP header, reassemble the packet and determine the destination. The packet will be routed and forwarded to its destination if its destination is not the local node. *Ip_forward()* will look up the routing table and find the corresponding network interface for this packet to be transmitted. *Dev_queue_xmit()* places this packet into the output queue of the corresponding network interface. Queuing disciplines are deployed here for network traffic scheduling. Within a queuing discipline, *hard_start_xmit()* is responsible for sending the scheduled packet through the network interface.

The current Linux kernel is equipped with many queuing discipline such as First In First Out, Random Early Detection, Class Based Queuing etc. the developers also can develop new queuing disciplines and integrate them into the kernel. In the Linux kernel, all queuing disciplines are managed by the traffic control suit (TC), which will be described in the next section.

3.3.2 Linux traffic control (TC)

Linux is an open source system. Its source code is free to public under the GNU License. Traffic control is a very important part for Linux based networks, which is responsible for all bandwidth sharing and packet scheduling issues. The Linux kernel input packet treatment process and output packet treatment process is given [WER01] [KUZ98]. Figure 3.22 displays the process.

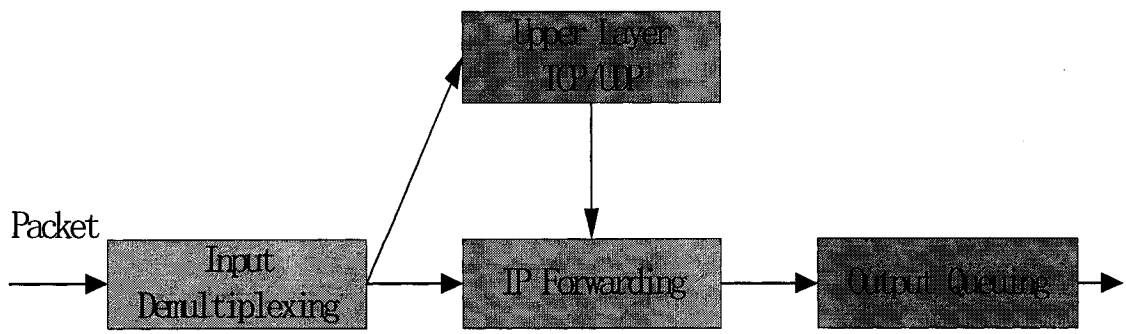


Figure 3.8 Packet treatment process inside kernel

The incoming traffic from the input interface is first demultiplexed. The demultiplexed packets are passed to different entities based on their destination address:

1). The forwarding entity when the destination address is not a local address. Its responsibilities are selecting the output interface, next hop, encapsulation etc. The forwarding entity is also responsible for handling the traffic generated locally.

2). The upper Layer entity when the destination address is the local address. Packets will be passed to higher layers for further treatment if the local node is the end system of a network application. The packet forwarded by the forwarding entity is queued to the output interface. This is where the traffic control exerts its function.

In the standard Linux kernel, all the traffic control happens in the output interface. The traffic control can decide the rule for packet queuing, packet dropping, packet transmitting sequence and packet delaying and so on. After traffic processing, the packet will be transmitted through the specified network interface directly. The traffic control can be configured with a user space control program, also known as shell script. By executing the script, the QoS related parameters are passed to the kernel in order to make specific traffic control function enabled.

The fundamental components of the Linux traffic control architecture are:

- Queuing Discipline
- Classes (queuing discipline identification)
- Filters/Policers/Classifiers

Queuing disciplines

Queuing discipline is embedded in each network interface. Its main function is to control how packets are enqueued on that particular interface. It applies certain policies to the packet passing this network interface, such as classified, be queued or dropped etc. First the Queuing discipline uses filters to identify the different types of traffic streams such as IP addresses or DSCP values, and classifies them into different classes.

The queuing discipline will process different classes differently, with different queue priority, different queue serving algorithms or rate etc. these parameters are important for allocating the network resource among different classes.

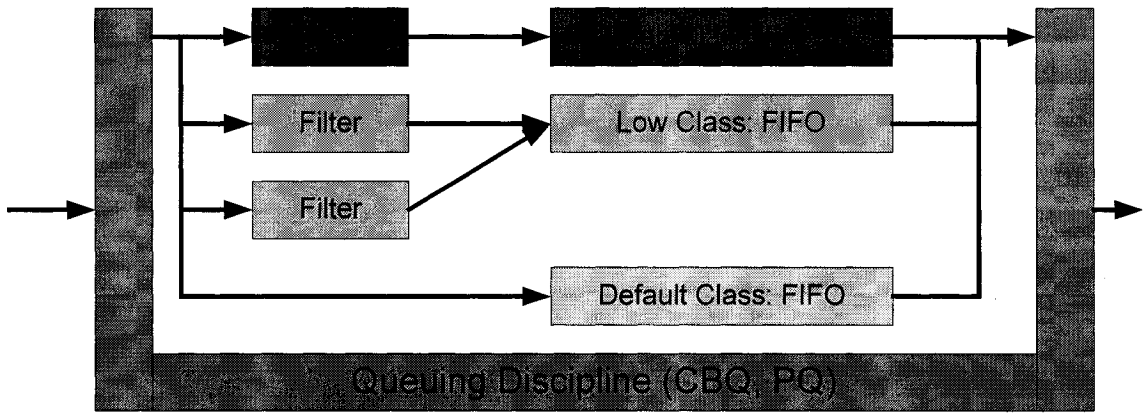


Figure 3.9 Queuing discipline with classes and filters

Fig 3.9 shows an example of a queuing discipline. Inside queuing discipline, incoming packets will be classified by a set of filters with different priorities; the simple FIFO queue then is then used in each class for processing the packets.

The queuing disciplines provided by current Linux kernel includes:

- Class Based Queuing (CBQ): This is Link-sharing algorithm for guaranteeing bandwidth requested by real time traffic and preventing the starvation phenomena among traffic classes.
- First In First Out (FIFO): The simplest queuing discipline. The incoming packet is stored at the end of queue if queue is not full; otherwise, the packet is dropped. The packet is dequeued by the packet arriving order.
- Token Bucket Filter (TBF): Control the packet transmitting rate and volume using tokens in the bucket (buffer). Often used in DiffServ for regulating the premium service traffic.

- Random Early Detection (RED): By using a queue length threshold to control the TCP stream transmission rate through dropping the packet for avoiding the traffic congestion.
- Generalized RED (GRED): Generalized form of the RED. Supporting different dropping priorities and buffer sharing.
- DSMARK: Supporting DiffServ. This queuing discipline can provide a framework for implementing the QoS among users.
- Priority: supporting multiple queues which have different transmission priorities. It is pre-emptive.

In order to control the packet, the queuing discipline provides the following supporting functions:

- Enquque(): Taking a packet from system. After taking a packet, queuing discipline places it into corresponding queue. Enqueue is invoked when a packet arrives at the output interface.
- Dequeue(): Dequeuing a packet for transmission. Dequeue is invoked when a packet leaves the system.
- Requeue(): Giving the packet second chance for transmission
- Init(): Initializing and configuring the queuing discipline
- Drop(): Dropping a packet from a queue
- Change(): Changing the configuration of queuing discipline
- Reset(): Setting the queuing discipline to the initial state
- Destroy(): Removing a queuing discipline and releasing the allocated resource
- Dump(): Returning diagnostic data for statistical use.

Classes

Classes and their functions are the fundamentals for the queuing disciplines. Normally there is one to one relationship between classes and queuing disciplines. Each class has its own queue for storing packets. Classes can be identified in two ways:

Filters/Policers/Classifiers

The main function of the filters is to classify the incoming packets into the corresponding filter's classes based on their properties e.g. IP address, IP protocol and port number during the enquque operation of the queuing discipline. Filters with the same property parameters of packets must have different priorities.

Filters also provide a set of functions for the control of the packets. These functions are:

- Classify: determining the class of an incoming packet.
- Init: initializing a filter
- Destroy: removing a filter.
- Get: looking up the filter list and returning the internal id of a filter.
- Change: changing the properties of a filter.
- Dump: returning the statistical data for a filter.

There are two types of filters in the current Linux kernel:

1) Generic filters: they are classifying incoming packets for all classes inside a queuing discipline.
    a. Cls_fw: Firewall based classifier; the packet is classified by the mark firewall assigned to the packet.
    b. Cls_route: Routing table based classifier; the packet is classified by using the routing rule in the routing table.

2) Specific filters: classifying incoming traffic only for one specific class
    a. Cls_rsvp: As mentioned in Chapter1, the RSVP protocol is used for message exchange for acquiring quality of service through dynamically reserving the network resources. The RSVP classifier is used for determining the packet's QoS class and treatment procedure.
    b. Cls_u32: often used in our test. The packet can be classified by various options, e.g. source or destination IP address, port number and protocol type etc.

3.3.3 Linux Differentiated Service

The basic structure of a DiffServ Node is described in Figure 3.10.

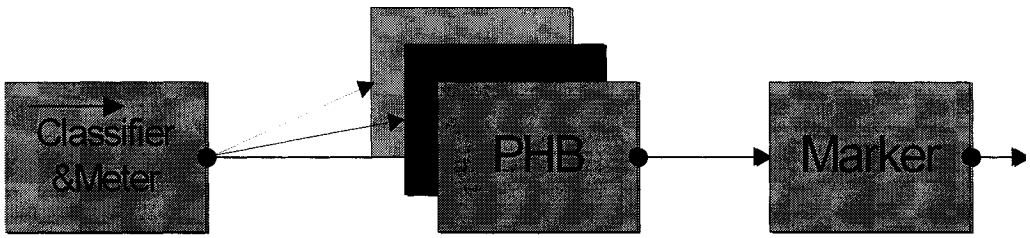


Fig 3.10 Differentiated Services Architecture

The sch_dsmark queuing discipline is responsible for setting tc_index, which is related to the traffic classes, and read the DSCP value to determine the service priority. It is the basis for the DiffServ framework with multi-operations. The PHB, as described above, is based on service priorities, e.g. Expedited Forwarding and Assured Forwarding, and can be implemented with Class-Based Queuing (CBQ) and General Random Early Detection (GRED), which has multi-dropping priorities.

Two types of classifiers are used in sch_dsmark framework:

- Cls_rsvp: It is used at the edge router and is responsible for marking the packet when it enters the DiffServ domain. After the cls_rsvp classification, the inner queuing discipline may read or modify the tc_index, and the DSCP value is set according to the tc_index. Figure 3.11 demonstrates the procedure.

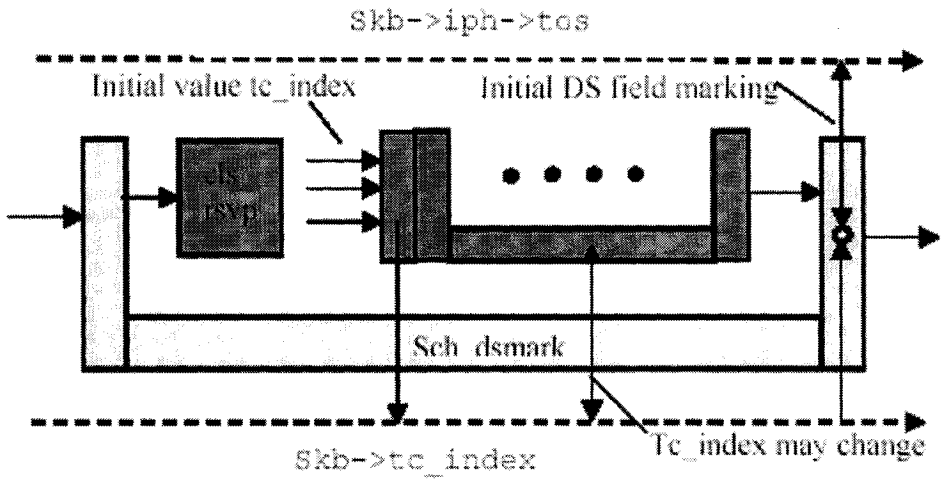


Figure 3.11 Micro-flow classifier (Marking) from [ADE02]

- Cls_tcindex: It is used at core router for handling the behavior aggregate by copying the DSCP value to the tc_index. Figure 3.12 demonstrates the procedure.

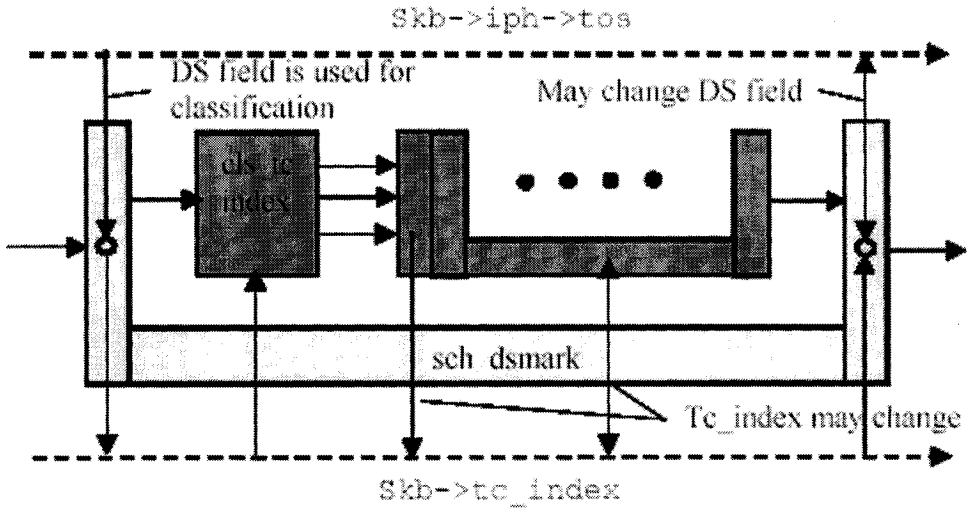


Figure 3.12 Behavior aggregate (PHB) classifier from [ADE02]

The Sch_dsmark queuing discipline uses above classifiers to extract the DCSP value from the incoming packet and send it to the inner queuing discipline for packet scheduling by the packet's service level, e.g. different PHB. After the inner queuing discipline is applied, the sch_dsmark sets the DSCP value and passes the packet to the network.

In order to control the DiffServ mechanism in Linux, we need to communicate and configure the kernel code or modules. In fact, the user level program *tc* provides us with the application interface for the above purpose. The interaction is shown in Figure 3.13.

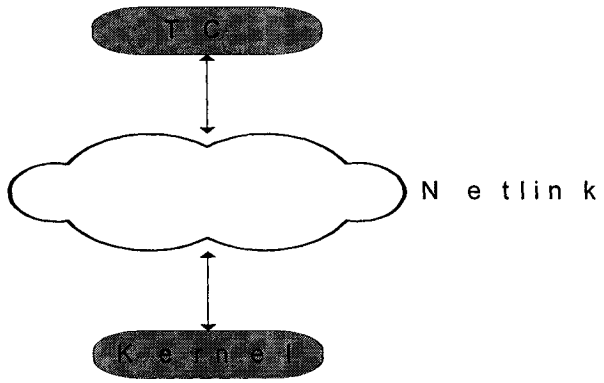


Fig 3.13 Communication between User space and Kernel

3.4 Summary

In this chapter, we have described the DiffServ architecture, ranging from the system's basic platform to system's components. In addition, we introduced the Linux operating system and Linux traffic control that is basis for implementing DiffServ over Linux. Important queuing disciplines for the support of the DiffServ architecture such as CBQ, PQ, RED etc has been described.

As an advanced QoS mechanism, DiffServ is more robust, simpler, and more scalable. With the spread of multimedia real time applications over Internet, the combination of Linux and DiffServ gives us a convenient QoS environment for deploying and testing real time applications over networks using a low cost and easily reconfigurable infrastructure.

4. Test-bed Configuration

4.1 DiffServ Testbed Configurations

The objective of study is to evaluate and understand the behavior of existing and emerging real time applications in a differentiated services environment, such as MPEG-2 video and DIVE, as well as uncover the mechanisms affecting the performance of such applications.

We have chosen for our analysis on experimental approach. By doing so, we are able to incorporate the behavior and limitations of real hardware/software processing units and protocol stacks in our analysis. This cannot be achieved through simulations.

Simulations are providing a low cost approach in assessing the behavior of a certain design, and in many cases (using proper assumptions and simulation techniques) are able to evaluate large and complex systems, which would require a major expense should an actual full scale implementation of the system had to be done. However, the simulator evaluates an environment that is the exact replica of the understanding developer has about the specific system. Unrealistic assumptions or oversight of potential processes and

impairments can provide results that are far from the reality. An evaluation based on a small size testbed allows to "see" the behavior of an actual system, and has a good chance to include if not all, at least most of the parameters, factors and impairments, which will be appearing during the operation of the final product. In many cases, such factors are difficult to be identified in advance and / or too many that is quite impossible to include all of them in a simulator of logic complexity. Coming to our case, a simulation would not have allowed to assess the impact, the sharing of CPU and other computing resources by multiple processes (of which several of them are not every part of the routing and packet forwarding functionality we are testing) has on the performance of the application, as it would have been difficult to impossible to model them accurately in the simulation. In addition, the implications of specific hardware used in the implication can be assessed. Thus, we achieved a more realistic evaluation environment, while its budget remained low.

The testbed configuration is divided into two parts:

- Hardware and software configuration
- Services implementation

4.1.1 DiffServ Testbed Topology

Figure 4.1 shows our DiffServ testbed network topology. The testbed comprises two DiffServ routers (Edge Router, Core Router), a video and a DIVE server, a video and DIVE client, three Ethernet segments, a traffic sink, two InterWatch 95000 (IW95000) network analysis/traffic generation tools manufactured by GN-Nettest. The video and virtual reality server hosts are located in the first Fast Ethernet segment 111.11.1.0/24 (100Mbps) and they are responsible for generating and sending the video and DIVE traffic into the network. One network generator is also located in this segment and is used to produce the background traffic. All traffic is multiplexed in this segment and injected into the same Fast Ethernet card of the edge router. The second Fast Ethernet network segment 111.11.3.0 (100Mbps) connects the output interface of the edge router and the input interface of the core router. A 10Mbps Ethernet network segment 111.11.2.0 is used

at the third network segment. As link speed is lower (only 10Mbps), it is capable of producing traffic congestion. The Video client, DIVE client and sink (background traffic receiver) are in this network segment.

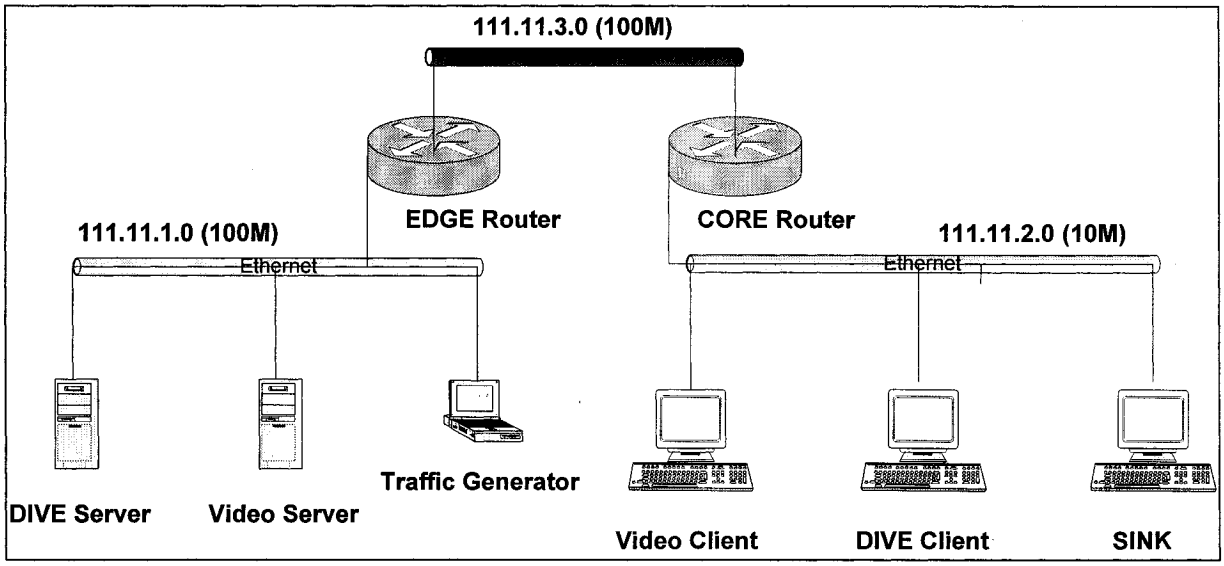


Figure 4.1 DiffServ Testbed Configurations

IP address as follows:

Table 4.1 IP distribution map

Host Name	IP Address
Video Server	111.11.1.2
DIVE Server	111.11.1.4
Traffic Generator	111.11.1.3
Edge Router	111.11.1.1 111.11.3.1
Core Router	111.11.3.2 111.11.2.1
Video Client	111.11.2.2
DIVE Client	111.11.2.4
Traffic Sink	111.11.2.9

4.1.2 Hardware and Software Environment

Because the I/O performance and CPU power of the router are important for the router's packet processing and forwarding mechanism, we selected high performance PC as our DiffServ routers (edge router and core router) in our testbed. The edge router is a Pentium III CPU 600MHz PC with 256MB RAM, two 3Com 3c905 network cards, running on the Red Hat Linux 7.0 operating system with kernel version 2.4.3. The core router is a Pentium III 1GHz PC with 512MB RAM, running on Red Hat Linux 7.0 with modified kernel version 2.4.3. The software used for implementing the DiffServ is iprouter2 plus TC open source package.

After compiling the Linux kernel with the TC and DiffServ module, the ordinary PC is functioning as DiffServ Routers. The QoS disciplines can be used for packet marking, forwarding and scheduling.

4.1.3 Video Server and Client

The video server is a Pentium II 300 MHz CPU, 128MB RAM PC with Red Hat 7.0. In order to process the video file quickly, two RAID hard disks are used for the storage of uncompressed and compressed video data. Considering the case of transporting high quality of video on demand to users, we used MPEG-2 in our evaluation. We do so because we believe that video on demand will be one of the main multimedia type of applications that will make extensive use of DiffServ network. Combined with multicasting, it would provide low cost high quality video on demand content to the general population.

- For MPEG2 video, the Video server streaming program *video_server* (developed by Mohamed Toukourou [TOM00]) was modified and implemented. The video server produces streaming video at the rate of 31 frame/second, it is using the Linux real time clock and applies the UDP protocol.

The video client is a Pentium III 600 MHz CPU, 256MB RAM PC with Red Hat Linux operating system version 7.0.

- In order to decode the MPEG2 streaming traffic quickly, a SCSI-II MPEG-2 video decoder card, product of Vela Inc, is installed in the client box. The decoder card can support MPEG-2 PS format (Program Stream) and TS format (Transport stream) with a resolution range from 352x240 pixels to 740x480 pixels. There is a 64KB on board memory as video data storage buffer for MPEG-2 data playback. The MPEG receiver sends the video data into a decoder card and the video file is saved for video quality evaluation based on PSNR.

4.1.4 DIVE Server/Client

As a pair of sender and receiver in our testbed, the DIVE server and client are dual genuine Intel 850 MHz CPU, 512 MB-RAM PCs with Windows 2000 operating system. In order to reducing the test complexity and leveraging convenience, a DIVE traffic generator, developed based on the DIVE traffic statistical model, is used for emulating the real DIVE operation [QIN99]. The traffic generator was developed using Winsock. It is for generating packets at the application layer. UDP protocol was used for its packet format. Client is used for receiving the DIVE traffic corresponding one or more DIVE server in network.

4.1.5 Network Apparatus – GN Nettest's IW95000

The GN Nettest IW95000 network test equipment is used in our testing for generating and monitoring the network traffic.

The GN Nettest IW95000 is a popular network-testing tool. Its testing capabilities cover ATM, Ethernet, MPLS, and optical networks. The GN Nettest IW95000 can provide valuable information about monitored traffic streams, such as traffic rate, link utilization, the captured packet's hardware address (MAC), IP address, used protocol and packet contents. In addition, the GN Nettest IW95000 can produce background traffic of different types, such as TCP, UDP, ATM, MPLS etc. the user has control of several parameters such as volume, distribution (e.g. constant bit rate or variable bite rates), protocols (UDP or TCP) etc.

4.1.6 Network Packet Analyzer (Software)

For the purpose of obtaining network performance related statistics, affecting the applications such as packet loss rate, average delay, delay jitter, a monitoring tool is required to be residing within the nodes capturing the incoming and outgoing packet for the purpose of capturing those parameters.

Instead of using *tcpdump,* an open source package with complex setup parameters, we developed a new traffic monitoring and capturing program *P-dump*. *P-dump* is developed with language C and is also based on the *pcap* library, which is used by *tcpdump*. However, *P-dump* has very a simple and friendly interface for user to setup capturing thread and process the captured traffic file.

4.1.7 Video compression and processing tool

In order to produce and evaluate MPEG based video streaming, several hardware and software are used. There are:

- **Video Capturing Card**: It captures analog video from videotapes and stores in M-JPEG format for further image processing.
- **BBmpeg**: This is an open source software package. Converting M-JPEG to MPEG2. the user has control over several parameters such as image size, bandwidth rate etc.
- **Modified Berkeley MPEG2 encoder/decoder**: We modified original MPEG2 encoder/decoder and added new features such as frame information output etc. It converts MPEG2 image into YUV format for PSNR calculation
- **PSNR calculation program**: It compares the decoded YUV format image with the original YUV format image to calculate the PSNR value. The MPEG2's video traffic entering the video client is passed to the video decoder, where the video file is regenerated. Lost packets are replaced with empty packets prior to the stream being passed to the decoder, thus the effect of the lost packets is reflected on the reconstructed video images. The images of the original and reproduced

video files are decoded as YUV pictures and are compared with each other. Thus, we can obtain the PSNR value.

- **Used Video Clip**

  For Mpeg2 video clip, we digitized and compressed segments from following movies ("Star-Wars", "Crouching Tiger and Hidden Dragon", "Animal World"). Video clips have been captured at a resolution of 320x240x30 frames per second, and encoded with a Q factor equal to 10. For "Star-Wars" stream, the observed average video transmission rate is close to 4 Mbps, while the observed maximum and minimum rates are 4.8 Mbps and 3.3 Mbps. For "Crouching Tiger and Hidden Dragon" stream, average rate is 2Mbps with peak rates 2.5Mbps and 1.5Mbps. For "Animal World" stream, average rate is 1Mbps with peak rates 1.25Mbps and 0.75Mbps.

4.2 Service Implementation

By Diffsev mechanism service definition and our testing objective, in our test-bed, premium service EF is used for supporting video and DIVE application, which offer a low loss, low latency and low jitter. Best Effort service is used for supporting background traffic.

4.2.1 Edge Router Service Configuration

In DiffServ, the edge router is responsible for packet shaping, policing and marking. In order to perform above functions, the edge router deploys MF (Multiple Field) classifier and TBF (Token Bucket Flow). In the MF classifier, the U32 filter is deployed for packet classification. The U32 filter is the most advanced filter available in the current Linux implementation. It has a robust behavior because of its hashing table based structure for discarding, forwarding at real time and prioritizing a packet. Packet classification can be performed based on source & destination IP address, source & destination TCP/UDP port, DSCP, TOS byte and protocol type. The DiffServ queuing Discipline *Sch_dsmark* is used for setting the DSCP according to an application service profile. In our case, we use

U32 to split the real time traffic (Video, DIVE) into Class 1:1, corresponding to packet DSCP value 0x2e (EF Class), the background traffic goes into class 1:2 as Default Class. See Figure 4.2.

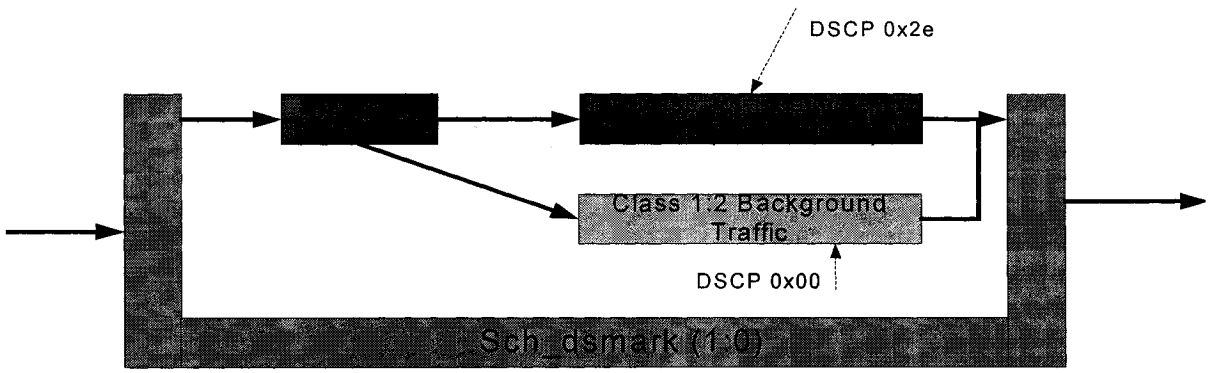


Figure 4.2 *Packet marking and classifying at edge router*

4.2.2 Core Router Service Configuration

The core router is responsible for forwarding packets, which come from edge routers, based on their aggregate behavior, DSCP value. By distinguishing the packet's DSCP value, the core router can decide to use corresponding queue scheduling to forward the packet.

In our case, Video and DIVE packets are marked with DSCP value 0x2e and are forwarded to the corresponding output queue Class 2:1 (fifo, VQ) with highest priority because of their EF PHB properties. Background traffic, with DSCP value 0x00, is forwarded to the output queue Class 2:2 (fifo) which has with lower priority. Figure 4.3 demonstrates the packet handling process inside the core router.

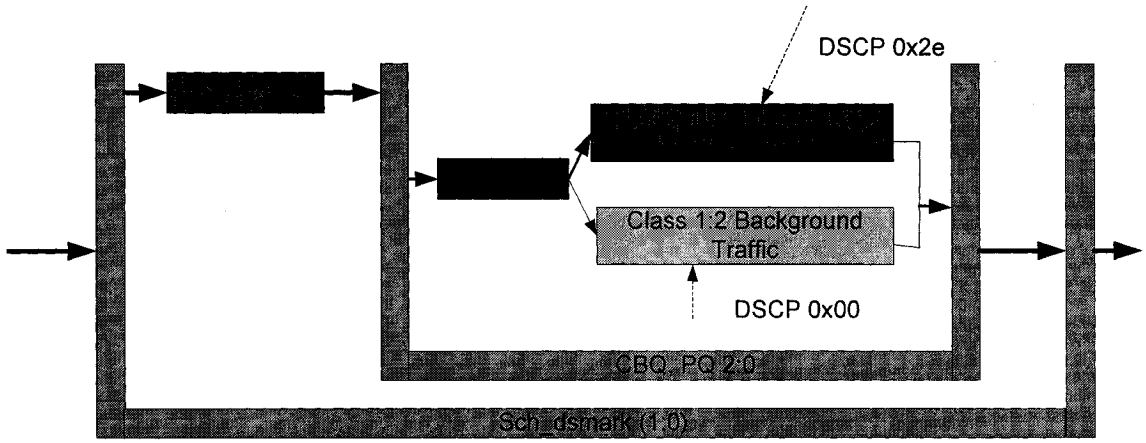


Figure 4.3 *Packet processing at core router*

4.3 Validation

In order to validate the effectiveness of the scheduling algorithm (PQ, CBQ) deployed in the DiffServ routers, we contacted a number of tests to verify the fundamental features of these algorithms. What we report here is the most meaningful data for verifying following feathers in our scenarios:

- Priority
- Service rate regulation
- Spare bandwidth sharing
- Flow isolation

The configuration used in test is shown in Figure 4.4. Host A and Host B are the source of traffic: EF, BE. Host C and SINK are the receivers of EF and BE traffic.

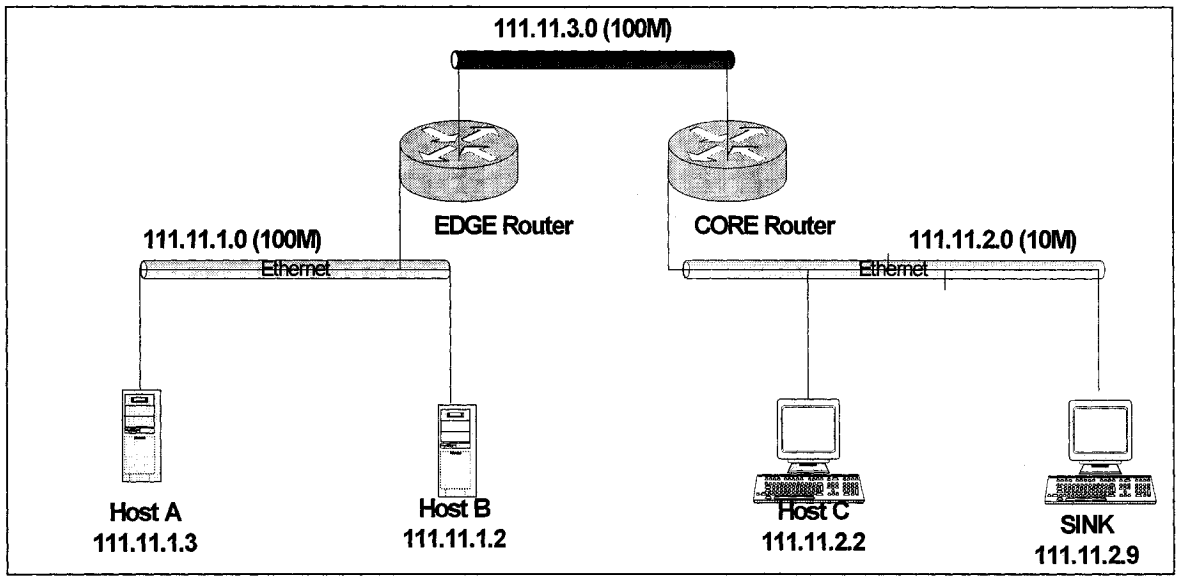


Figure 4.4 Test Bed for scheduling algorithm validation

1) Priority Queuing validation

In this scenario, as described above, two flows are used for testing: EF traffic running at 6Mbit/s, BE running at 10Mbit/s, with packet size 1500 bytes.

The scheduler in DiffServ core router was configured as PQ, EF traffic as high priority traffic and BE traffic as low priority traffic. The results are shown in Figure. 4.5.

Each traffic flow starts the transmission at different times. In the first stage, the high priority EF traffic is present with 6Mbit/s bandwidth occupation. When the low priority BE traffic is present with 10Mbit/s, the EF traffic still maintains its 6M bit/s traffic rate. When EF traffic is not present, BE traffic takes all the bandwidth 10Mbit/s. When EF traffic is present again, it takes the bandwidth back from BE traffic. This indicates that PQ scheduler take control of the bandwidth and let EF traffic pass first. From the figure, we can say that the results are satisfying for validating the PQ scheduler under DiffServ.

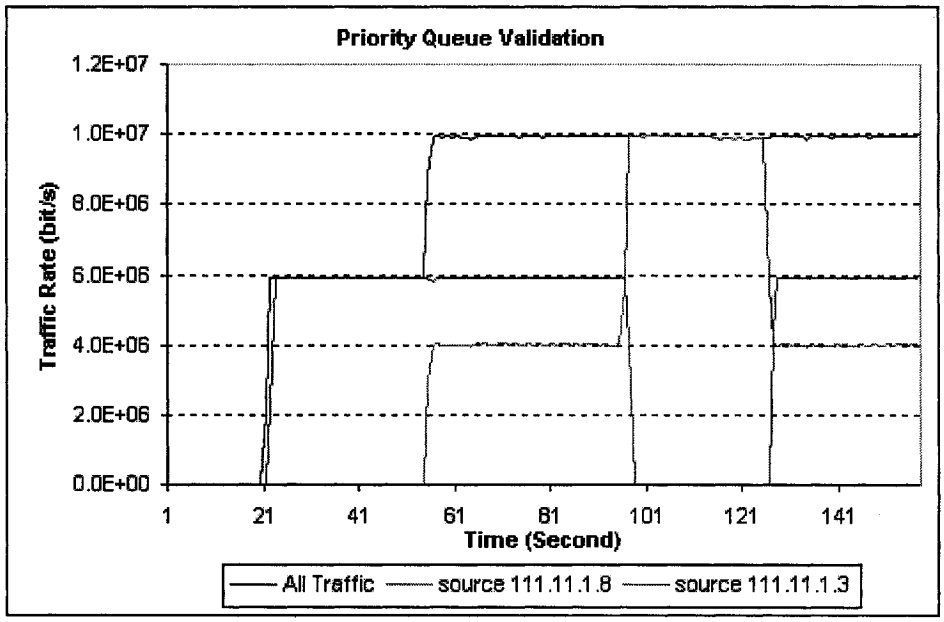


Figure 4.5 Traffic priority under Priority Queue.

2) CBQ validation

In this scenario, two flows are used for testing: EF traffic running at 6Mbit/s, BE running at 10Mbit/s, with packet size 1500 bytes.

The scheduler in DiffServ core router was configured as CBQ, EF traffic and BE Traffic have the service rate limit 1Mbit/s and 9Mbit/s. the results are shown in Figure. 4.6.

Each traffic flow starts the transmission at different times. In the first stage, the high priority EF traffic is present and take all the bandwidth it needs: 6Mbit/s. When the BE traffic is present with 10Mbit/s, the spare bandwidth is shared proportionally based on the bandwidth allocation profile: EF traffic 1M bit/s and BE 9Mbit/s. When BE traffic is not present, EF traffic takes all the bandwidth it needs again. This indicates that CBQ scheduler take control of the bandwidth allocation and let different traffic pass through the core router with pre-defined bandwidth. From the figure, we can say that the results are convincing for validating the CBQ scheduler under DiffServ.

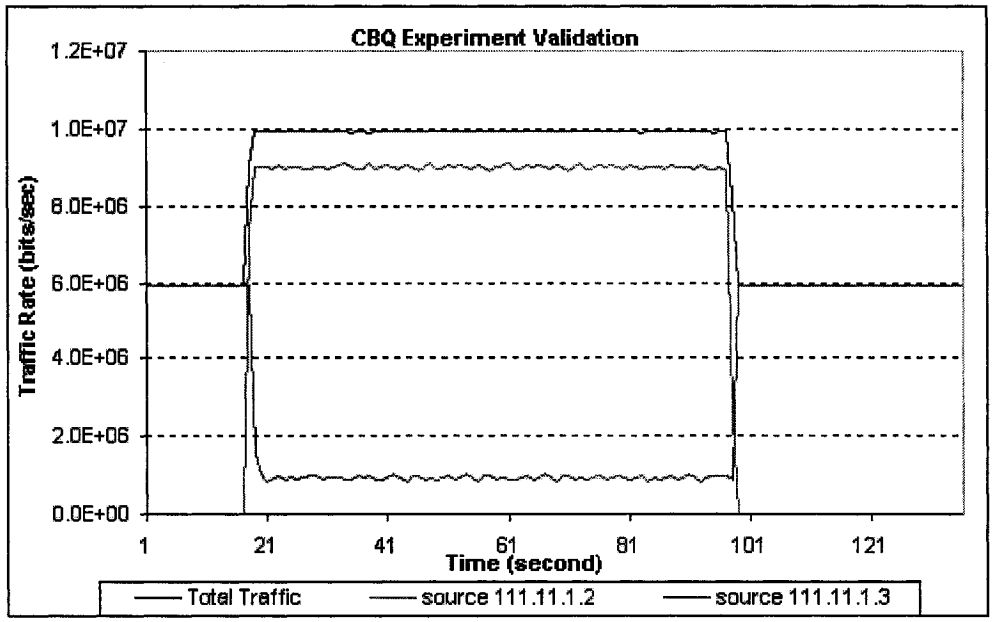


Figure 4.6 Spare bandwidth sharing and service rate limit under CBQ.

4.4 Summary

In this chapter, we present the DiffServ testbed hardware topology and software implementation, as well as the software tools we used for collecting and analyzing these experimental data. At the same time, the DiffServ edge and core routers implementation also are introduced in order to be able to understand the packet handling process inside the DiffServ architecture. In the end, we introduce the procedure of DiffServ scheduling algorithm validation and provide the standard for further testing.

5. Test Results Analysis

5.1 Overview

In this chapter, we present and analyze our experimental results. As mentioned in Chapter 1, the objective is to evaluate the performance of Video and DIVE in a best effort and DiffServ-enabled IP network and to explore the DiffServ architecture QoS properties of supporting the mixed traffic composed of real-time traffic and non-real-time traffic.

In the test-bed, the core router output interface becomes the traffic bottleneck, because of its 10Mbps output. In order to protect the real-time traffic against heavy background traffic, the DiffServ mechanism is embedded into the core router.

Two typical real-time applications, Video and DIVE, are used and evaluated. The results are presented in the following sections.

5.2 Mpeg Video Application

5.2.1 Mpeg2 Application

In order to evaluate the MPEG2 video application over DiffServ-enabled IP networks, the PSNR (Peak Signal to Noise Ratio) index is used for assessing the quality of received MPEG2 video. Fig 5.1 shows the process of calculating the PSNR.

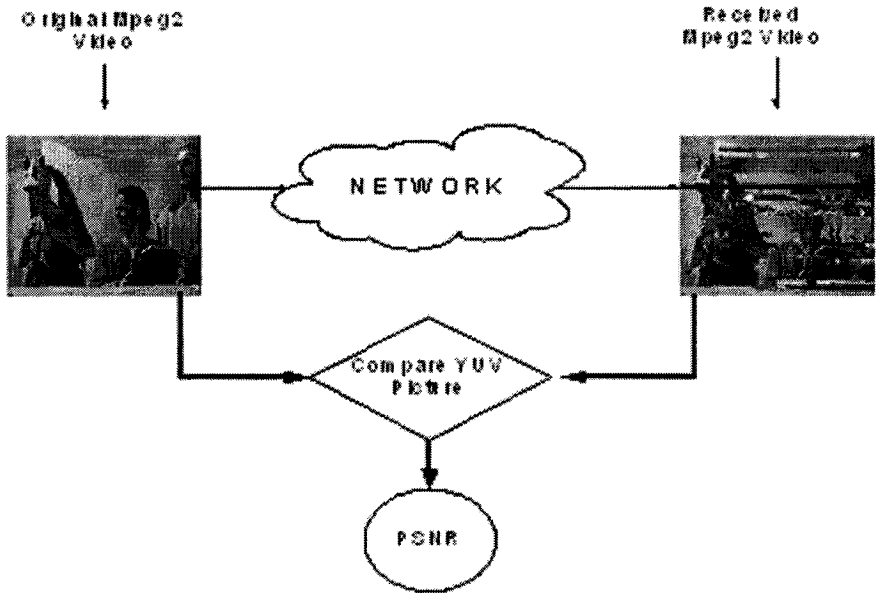


Figure 5.1 PSNR calculation procedure

At the receiver side, every image frame is reconstructed based on the received packet contents. The PSNR value can be obtained by comparing the reconstructed image frame and the corresponding original image frame using formula (2-3). From Figure 5.1, we can see that the received image is distorted and even hard for viewing compared to the original image. The calculated PSNR value this is 15.625478 dB. When the reconstructed image frame is identical to the original image, the PSNR becomes infinity. For our results limit the maximum value of PSNR is 60dB.

As mentioned in Chapter 2, two parameters, packet loss and packet delay in the network layer are crucial to the performance of real time applications. As the packet loss and delay increase, the quality of video decreases, as well as PSNR value.

For better understanding the Mpeg2 video application performance, we also used the frame inter-arrival time as the index for assessing the quality of video. Under ideal conditions, the received video frames should be played at the receiver side at the same speed as they are generated at the sender. However, the queuing delays of network delay video packets that are needed to reconstruct the image frame at receiver, thus affecting the video playback at the receiver side. This reflects at the frame inter-arrival time. Through the frame inter-arrival time, we can obtain the jitter of the received frames. The larger the jitter is, the lower the quality of the received video. For acquiring the frame inter-arrival time, the start time and the end time of a received video frame should be identified. As we can put the frame information into the contents of video packets at the sender side, when the video packets are received at the client's side, the video frame they belonging to can be identified. Since all arriving video packets are in order through the one-way backbone testbed, we can obtain the frame inter arrival-time by identifying two consecutively arriving video packets that belong to different video frames. The combination of the frame inter-arrival time and the PSNR index allow us assess the quality of video more accurately [FEA02].

After going through several trials with different clips from the movies mentioned in Chapter 4, we selected a 5-minute extract from the movie *"Star Wars"*. The clip has been captured at a resolution of 320x240x30 frames per second, and encoded with a Q factor equal to 10. The observed average video transmission rate is close to 4 Mbps with maximum and minimum rates 4.8 Mbps and 3.3 Mbps. Because of the high level of activity and fast scenes changing (Generating the traffic burstiness), it is suitable for using to evaluate the performance of network. The burstiness level of the video clip is 1.25, which represents a scenario that resembles better traffic aggregation, because in the core router of network, there is an aggregation of traffic streams. In addition, the peaks of the video traffic is not over more than 50% of the link capacity, thus it will not be monopolizing the resource during peak rate periods.

In order to see how DiffServ affects the performance of network and improves the quality of video (PSNR), we conducted the following tests under different network conditions:

1. Testing under emulation of experiment
2. Best Effort Experimental Test
3. DiffServ Experimental Test

5.2.2 MPEG2 Video Performance Test under Best Effort (Emulation)

Objectives of the experiments conducted and reported in this section are to provide an understanding, how specific packet losses and the distribution of their occurrence affect the performance of MPEG2 video. Such analysis will allow us to understand how error rate and error duration are affecting the quantities we use to measure the video performance.

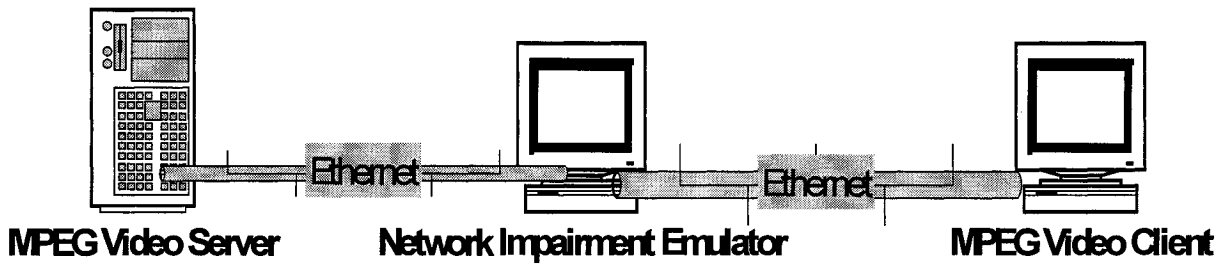


Figure 5.2 Experimental test-bed for MPEG analysis

The experimental set-up used for this purpose is shown in Fig. 5.2. Video packets generated at the video server are passed through the "Network Impairment Emulator". The network impairment emulator destroys video packets according to certain policies. The remaining packets are passed directly to the "client", where they are processed for the reconstruction of the original frame.

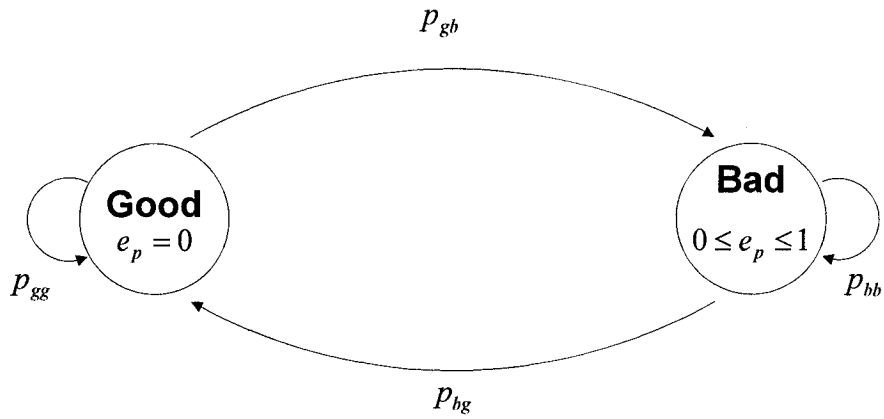


Figure 5.3 Impairment Emulator Model

Packet losses are generating according to a two-state model, shown in Fig 5.3. Let us define the probability that a packet is destroyed as probability e_p, $(0 \le e_p \le 1)$. When in the Good state, all packets are delivered to the client probability $(e_p = 0)$. However in the Bad state, probability e_p is different from zero. Two more parameters of the model are the time in the model are the times τ_g and τ_b. τ_g corresponds to the time the system spends in the Good state; τ_b in the Bad state. During the experiments, the following video related parameters are monitored and recorded:

- PSNR of the received frames
- Number of lost I, P and B frames
- Video frame interarrival times at the client.
- MOS test ratio of the received video clip.

Every experiment is repeated 8 times and every time, the 5 minutes video was run at its full length. In the first set of experiments, the value of τ_g and τ_b are set to certain deterministic values. These are:

- $\tau_g = 10$ ms, $\tau_b = 10$ ms
- $\tau_g = 50$ ms, $\tau_b = 50$ ms

- $\tau_g = 100$ ms, $\tau_b = 100$ ms

In all cases, τ_g and τ_b are of equal size. The objective is to assess the impact that the length of the duration of staying at a good or bad state has on the performance. The first case corresponds to Good and Bad periods that have approximately the duration of 1/3 of the frame generation period (33 ms). The second case exceeds this period. The last spans over more than three frame periods, thus generating a more "time persistent" impairment effect.

The experiments are performed for the following values of average packet loss rates P_{vl}:

- $P_{vl} = 10^{-1}$
- $P_{vl} = 10^{-2}$
- $P_{vl} = 10^{-3}$
- $P_{vl} = 10^{-4}$

In the case of deterministic τ_g and τ_b, P_{vl} and e_p are related as follows:

$$e_p = \{(\tau_b+\tau_g)/\tau_b\} P_{vl} \qquad \text{(Eq. 5.1).}$$

For each set of experiments, we calculate and present the following:

Average PSNR with lost frames included with the value 0: The PSNR value of a lost frame is set to zero. Then, the average PSNR of the video click run is determined. The determined average PSNR vales of all runs of the video clip are averaged and the final value of *Average PSNR* is produced.

Standard Deviation of PSNR with lost frames included with the value 0: The process is similar but, in this case performed for the standard deviation. For each "run" of the click, the standard deviation of the PSNR is determined. The standard deviations of all the repeated individual experiments are averaged and the final value is produced.

Average PSNR with lost frames not included: All lost frames are excluded. Then, the average PSNR is calculated from the remaining video frames. The average PSNR values of all runs of the video clip are averaged and the final value of *Average PSNR* is produced.

Standard Deviation of PSNR with lost frames not included: The process of calculation is the same with the calculation of PSNR standard deviation mentioned above, the only difference being that the lost frames are excluded.

Average number of I, P and B frames: The number of lost I, P and B frames in each run is determined. The average, corresponding to all the runs is calculated.

Average MOS rate: For each run, the MOS rate is determined. The average, corresponding to all the runs is calculated.

As indicated, two different cases of PSNR are considered. In the first case, lost frames are included in the calculations with frame PSNR value zero (0), whereas in the second they are omitted. The purpose for this distinction is to assess the quality of frames that are recoverable in terms of PSNR, as well as the overall quality of the video.

In figures 5.4 and 5.5 below, the PSNR values recorded in three different runs versus the total frame numbers received at client are shown. Each figure corresponds to a different set of τ_g and τ_b. Also, each figure presents results for $P_{vl} = 10^{-1}$ and $P_{vl} = 10^{-2}$.

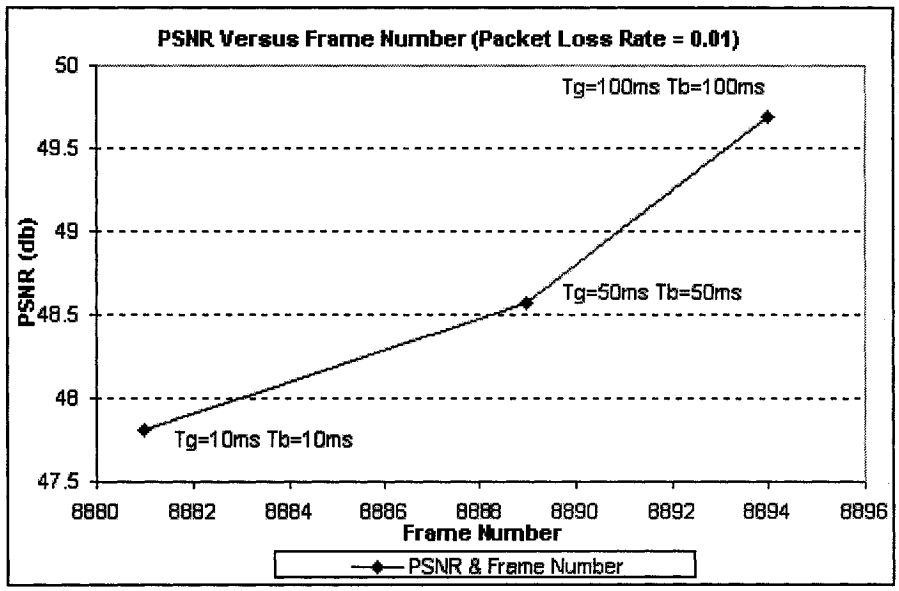


Figure 5.4 PSNR versus Frame Number ($P_{vl} = 10^{-2}$)

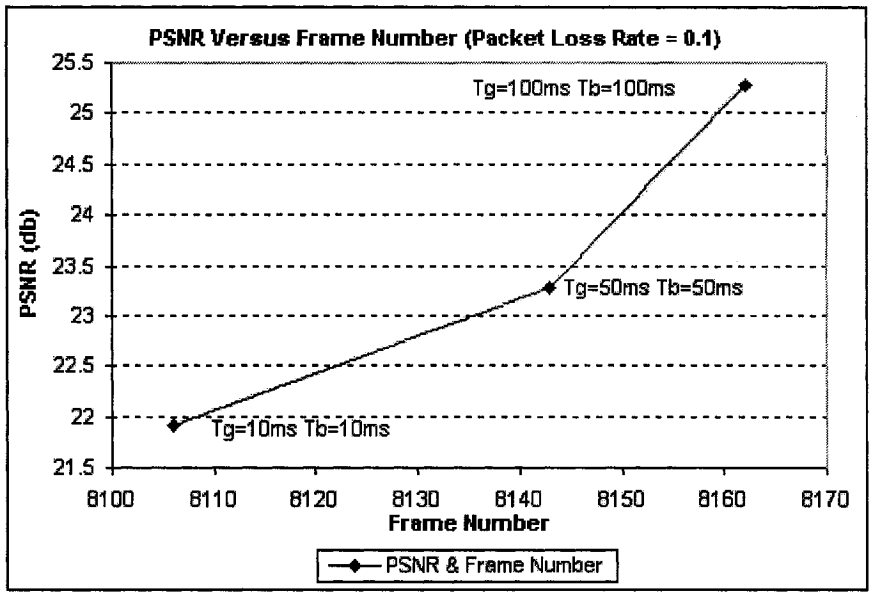


Figure 5.5 PSNR versus Frame Number (for $P_{vl} = 10^{-1}$)

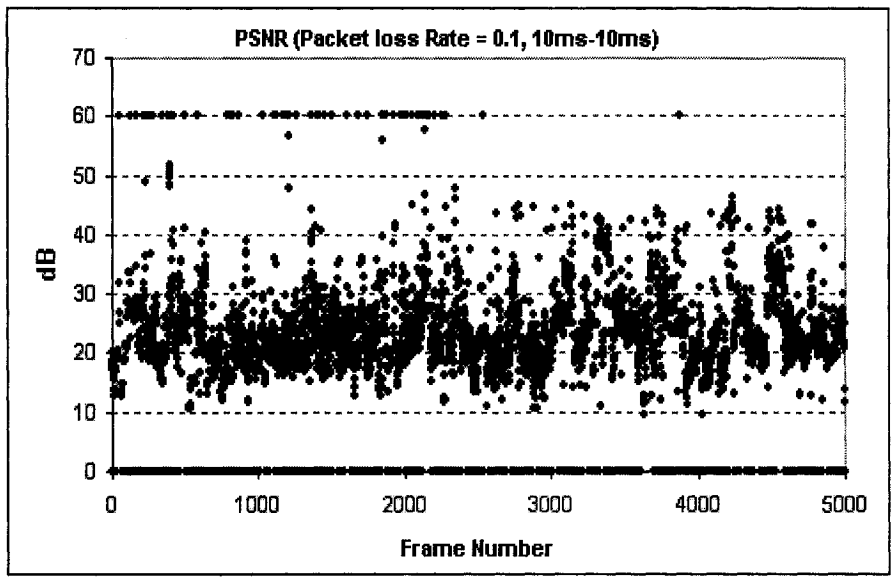


Figure 5.5a Frame PSNR graph for $\tau_g = 10$ ms, $\tau_b = 10$ ms with $P_{vl} = 10^{-1}$

Figure 5.5a presents the frame based PSNR graph. Because of high packet loss rate, most frames are damaged or lost, the PSBR value is dropped to 0dB or scattered between 10dB and 40dB. The average PSNR value as shown in figure 5.5 is 21.90881dB.

Figure 5.6 provides the average PSNR value for the three cases of packet loss described earlier, as a function of the average packet loss rate. The displayed values of average

PSNR correspond to when the lost frames are included with PSNR value zero. From the figure, it appears that shorter bursts of losses produce slightly worse performance. However, for all practical purposes, the quality of video, as is expressed through the average value of PSNR appears to be insensitive to the duration of the burst.

In the following figure 5.7, the standard deviation of PSNR is shown. Again, all cases appear very close to each other. However, it is interesting to observe the low value of standard deviation experienced for $P_{vl} = 10^{-1}$ as compared to $P_{vl} = 10^{-2}$. This is due to the fact that quite a number of frames in the $P_{vl} = 10^{-1}$ case are lost, thus providing directly set value of zero.

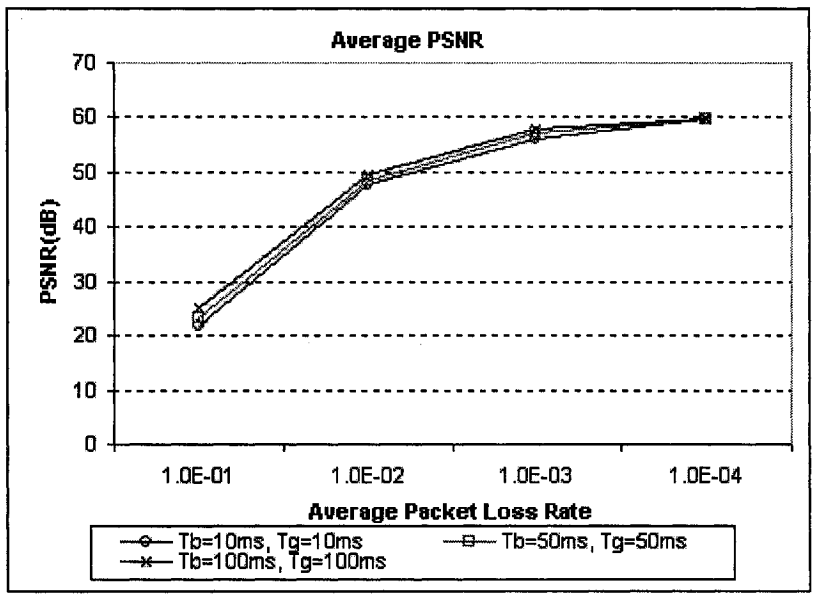


Figure 5.6 Average PSNR versus Packet Loss Rate when the lost frames are included in the calculation

Figures 5.8 and 5.9 are providing the average PSNR and standard deviation values, when the lost frames are not included. Comparing with figures 5.5, 5.6, it is evident there is noticeable difference in the value of PSNR only for the case of high packet loss rate. For $P_{vl} = 10^{-1}$ we see an increase in the value of average PSNR close to 8 dB. There is a very slight improvement for $P_{vl} = 10^{-2}$ (less than 1 dB), while for the rest of the cases it is

unnoticeable. When comparing the figures corresponding to the standard deviations, we observe it is lower when the lost frames are excluded from the calculation.

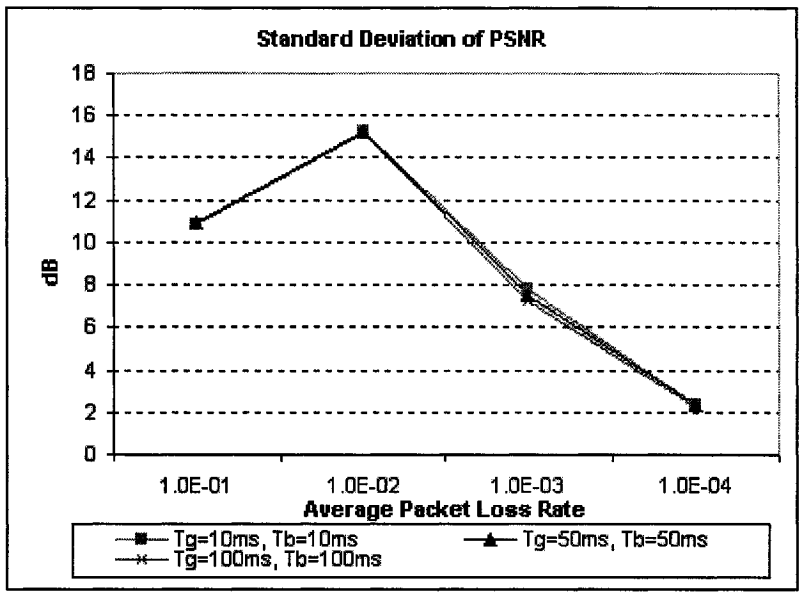


Figure 5.7 Standard deviation of PSNR versus Packet Loss Rate when the lost frames are included in the calculation

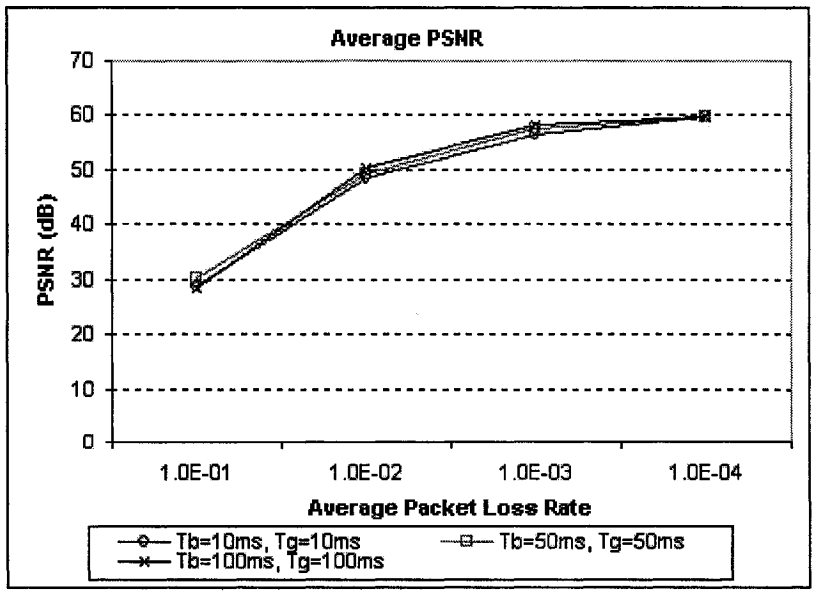


Figure 5.8 Average PSNR when the lost frames are not included in the calculation

The two following figures 5.10, 5.11 provide the average frame interarrival time and its standard deviation versus the packet loss rate, for the three combinations of τ_g and τ_b we are examining. The interarrival time frames are produced is 33.33 ms. According to the results, for high packet loss rates, the frame interarrival times could increase considerably. This is due to the complete loss of quite a number of frames (which increases the inter-arrival time measured). The same holds for the standard deviation as well. However, for packet losses below 10^{-2}, there is no noticeable effect. From the results it is evident that the packet loss durations are those producing worse results.

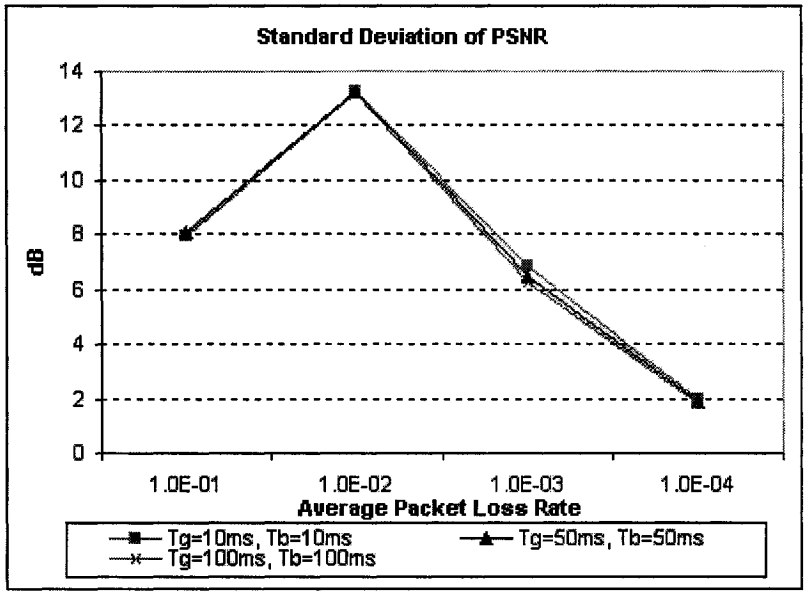


Figure 5.9 Standard Deviation of PSNR when lost frames are not included in the calculation

Figure 5.12 provides the average frame loss rate. It is evident that the frame is suffered most when packet loss rate is increasing. More packets loss means more frames loss because the frame is composed of packets and the frame header information is encapsulated in the payload of one packet.

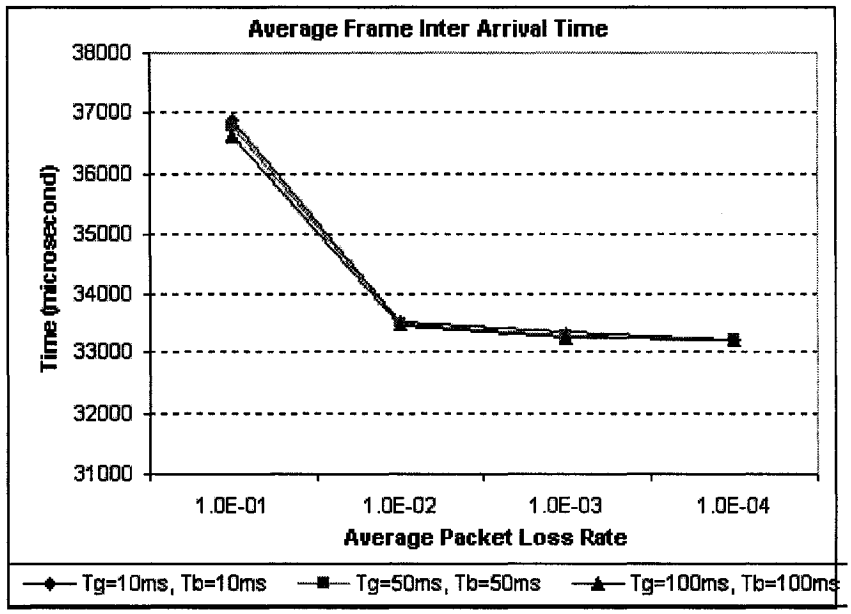


Figure 5.10 Average Frame Interarrival Time

In the meantime, we use another video quality index MOS (Mean Opinion Score) to evaluate the received video performance. Like PSNR value, with the packet loss probability decreasing, the MOS value is changed from 2.1 (annoying) to 4.8 (Almost imperceptible), the video performance becomes better, as shown in Fig 5.13.

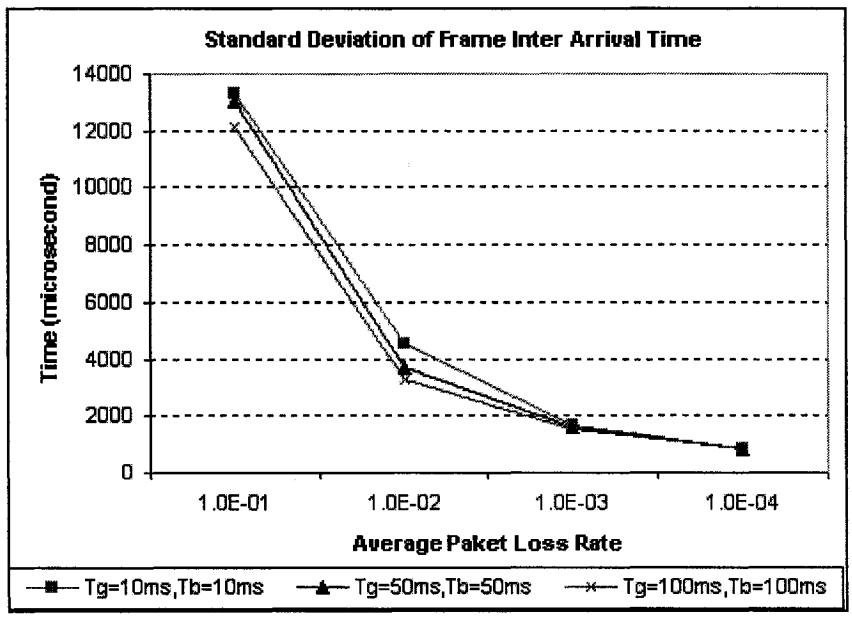


Figure 5.11 Standard Deviation of Frame Inter Arrival Time

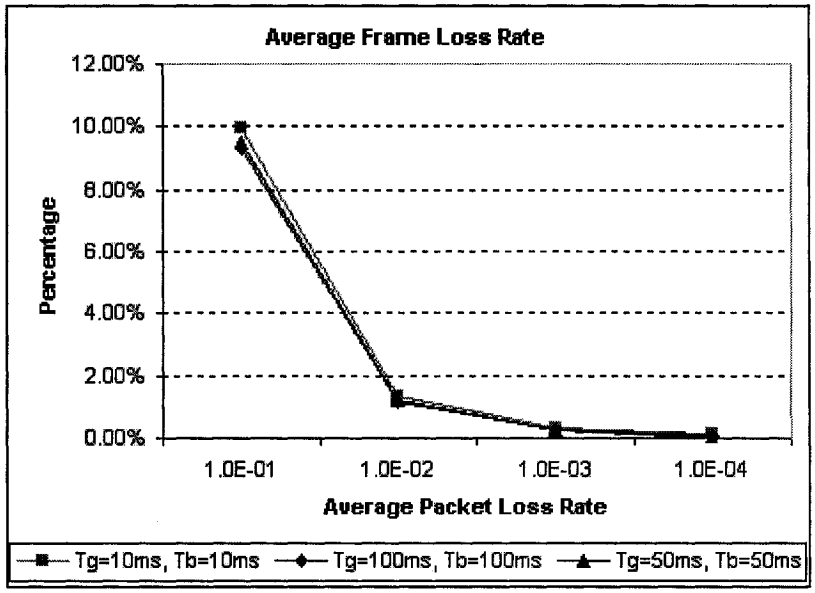


Figure 5.12 Average Frame Loss Rate graph

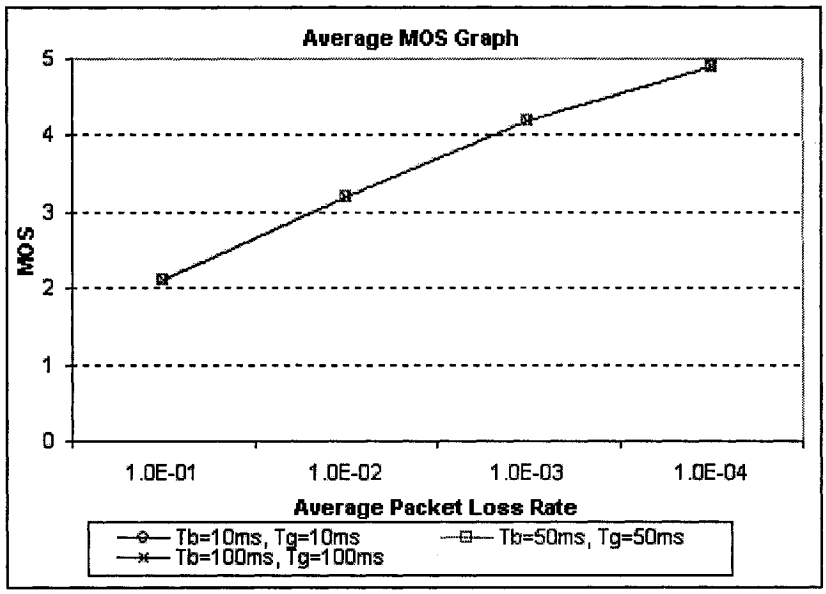


Figure 5.13 Average MOS graph

A second set of experiments has been contacted to assess the impact of dissimilar Good and Bad periods. Similar experiments with those described earlier have been contacted and the acquired results are compared, for the following cases:

- $\tau_g = 50$ ms, $\tau_b = 10$ ms

- τ_g = 50 ms, τ_b = 100 ms
- τ_g = 2 ms, τ_b = 240000 ms

Please note the last case is close to the scenario where losses are occurring during the entire period of the video transmission.

Figures 5.14, 5.16 display the average PSNR and its standard deviation versus packet loss rate for the above cases. Lost frames have been included with PSNR value zero. Figures 5.15, 5.17 also display the average PSNR and its standard deviation when lost frames are not included in the calculations. We see the increase of PSNR value when lost frames (PSNR=0) are not included in the calculation. However, the changing trend of PSNR value remains similar. Another interesting phenomena we noticed is that in standard deviation figure, all data go down for error rate 0.1 except the τ_g = 50 ms, τ_b = 10 ms. The reason is, the PSNR value for τ_g = 50 ms, τ_b = 10 ms case is around 40dB, the is the point where PSNR value varies most, some in very good (60dB) and some in very bad (0 dB). For other cases, only for error rate 0.01, PSNR variation reaches its most.

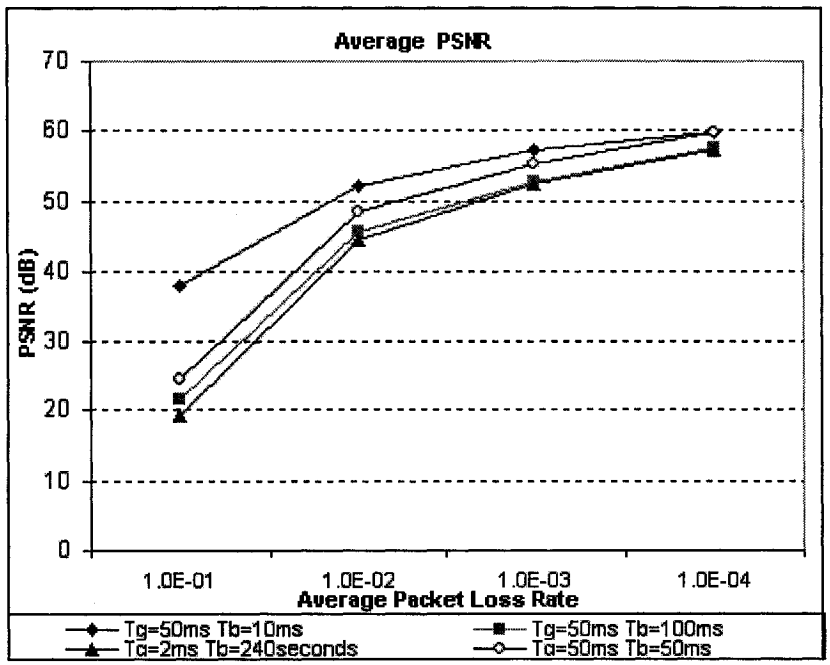


Figure 5.14 Average PSNR versus Packet Loss Rate when lost frames are included in the calculation.

It is evident that with the long duration of the bad state, the video quality presented worse performance as expected in PSNR value (Fig 5.14, Fig 5.16) and in frame loss rate 5.20 PSNR value varies much bigger when comparing the case τ_g = 2 ms, τ_b = 240000 ms with the case τ_g = 50 ms, τ_b = 10 ms.

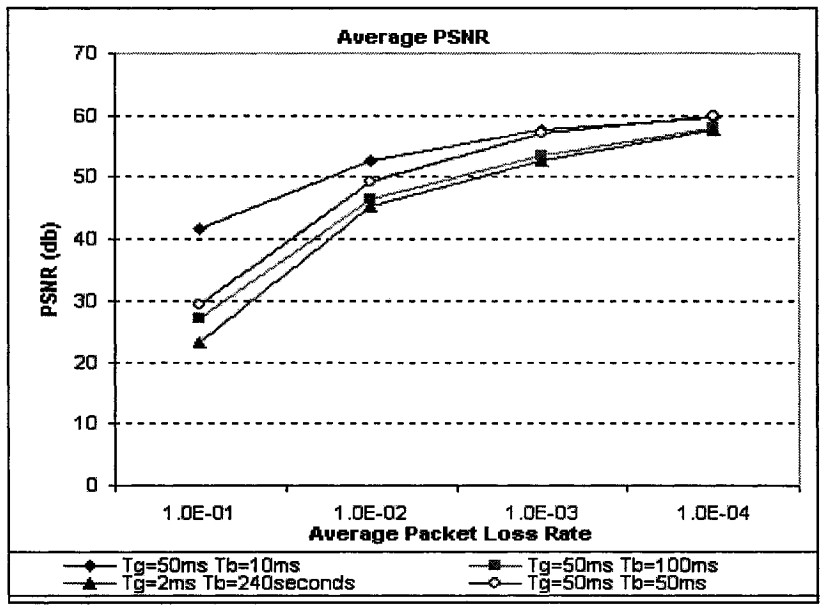


Figure 5.15 Average PSNR versus Packet Loss Rate when lost frames are not included in the calculation.

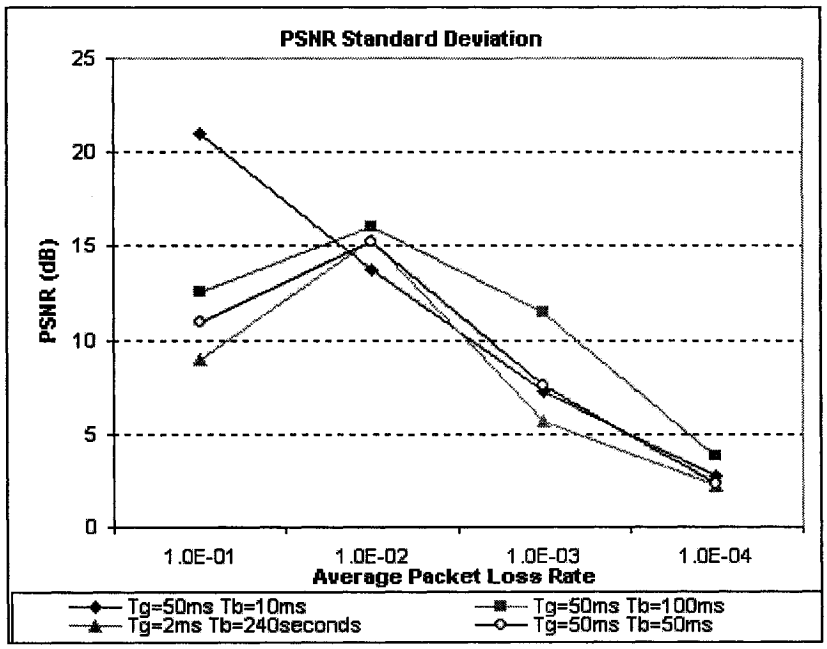


Figure 5.16 Average Frame PSNR standard deviation graph

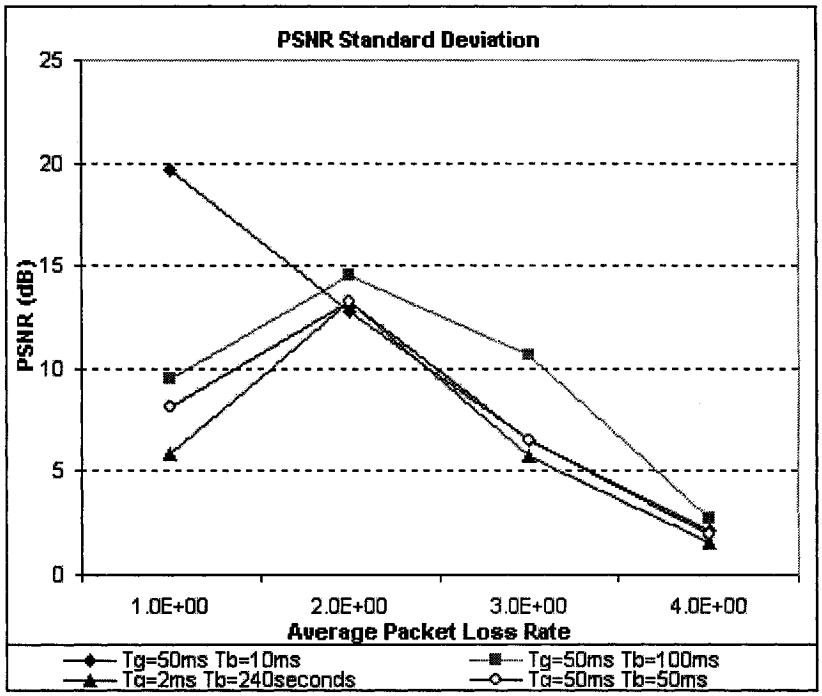


Figure 5.17 Standard Deviation of received Frame PSNR standard deviation

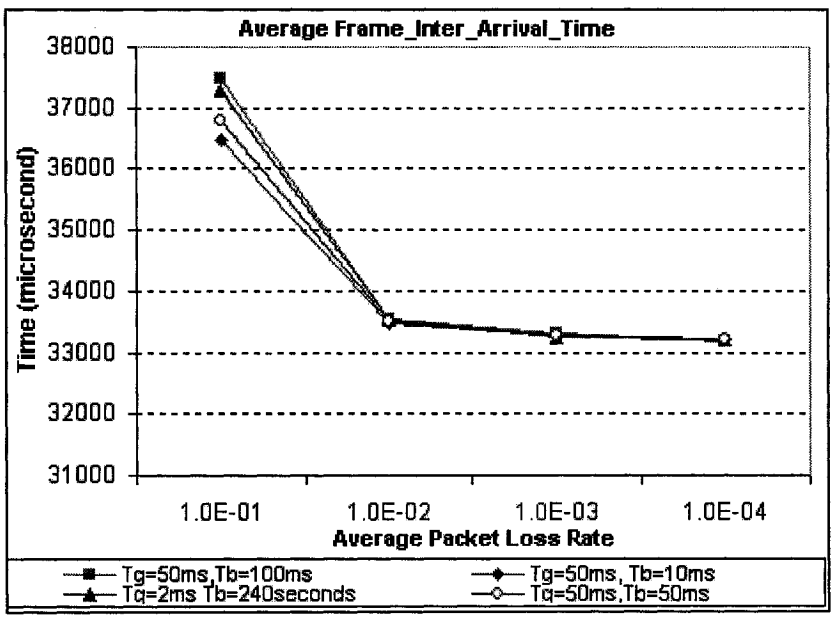


Figure 5.18 Average Frame Inter Arrival Time

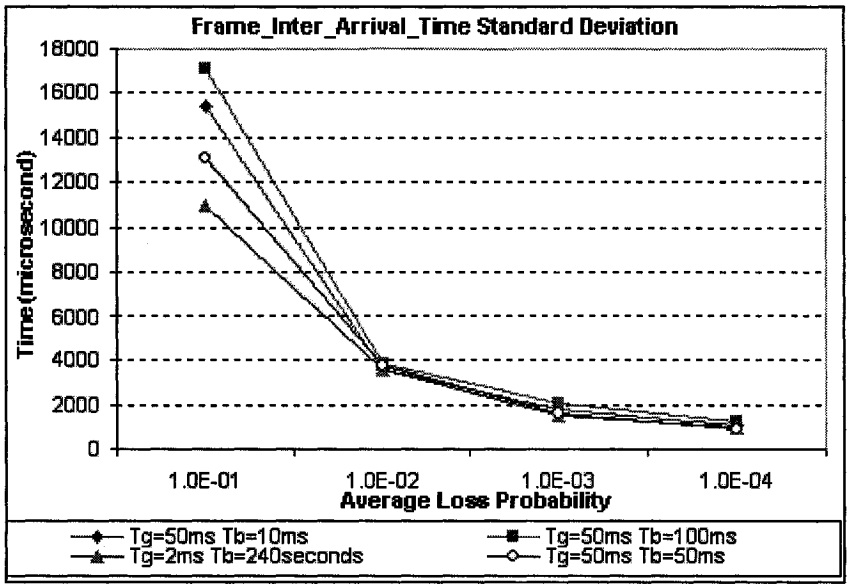


Figure 5.19 Standard Deviation of Frame Inter Arrival Time

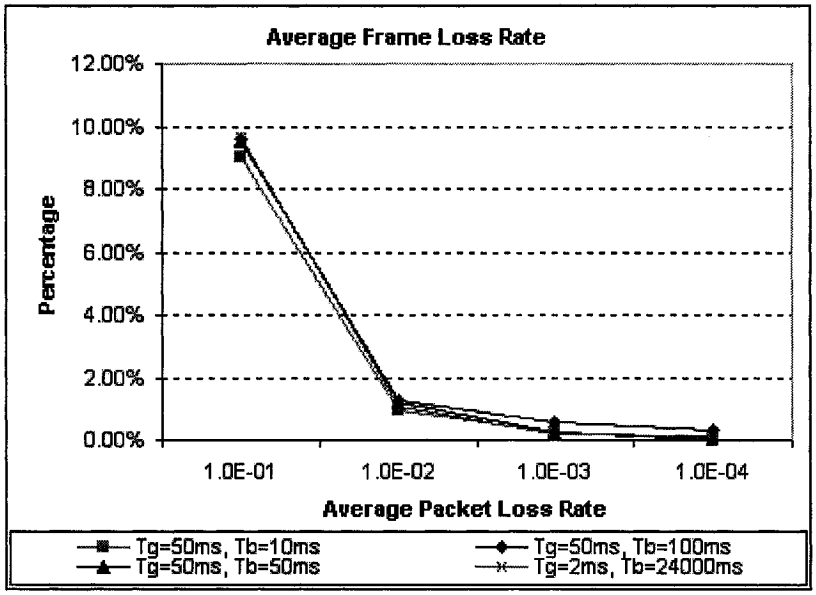


Figure 5.20 Frame Loss Rate

5.2.3 Video Performance under Best Effort

In this section, the performance of MPEG-2 streamed video in a best effort environment is assessed through experimentation. The results will be used and compared with those of the Differentiated Services architecture.

The network architecture of out testbed is shown in figure 5.21.

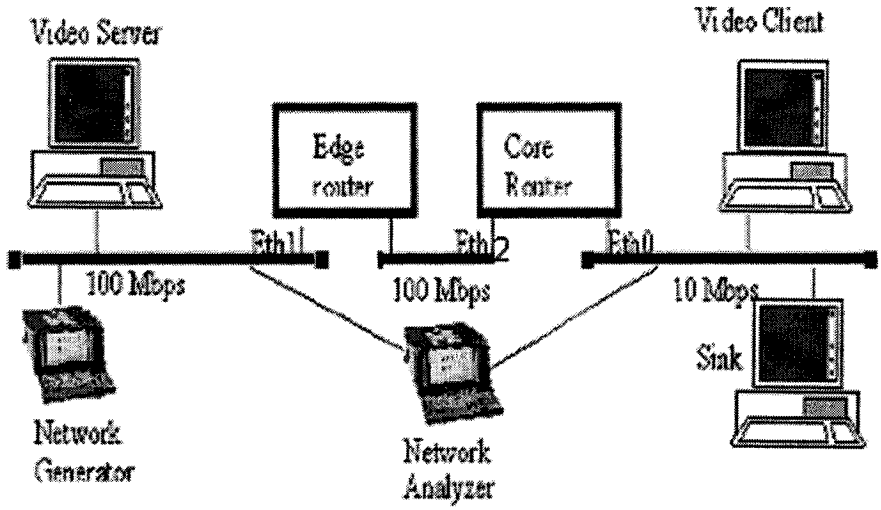


Figure 5.21 Experiment test-bed network architecture

In this case, the core router is operating on the best effort basis. The network generator is used to produce background traffic, which is multiplexed on the Fast Ethernet bus of the "Eth 1" network with the video traffic produced through the server. The combined traffic is passed to the Edge router, which in this case does nothing more but to pass it to the "Eth 2" segment. From there, the traffic is forwarded to the core router, which now operates as best effort router. The output of this router is linked on an "Eth 0", which is an Ethernet segment (10 Mbps), thus limiting the available bandwidth, generating bottleneck. The video client and "Sink" are receiving the video and background traffic respectively.

Experiments are contacted with the GN Nettest traffic generator producing Ethernet frames of 1500 bytes size. All experiments are conducted for the following background traffic loads:

- 0 Mbps
- 2 Mbps
- 4 Mbps
- 6 Mbps
- 8 Mbps
- 10 Mbps

We are conducting the experiments for three different values of buffer size at the core router:

- 5 packets (7.5kbytes)
- 10 packets (15Kbytes)
- 100 packets (150Kbytes)
- 1000 packets (1500Kbytes)

The following parameters are measured:

- Packet loss rate at core router
- Forwarding delay at the core router
- Number of video frames lost
- PSNR value of the individual frames
- Buffer occupancy at client side.
- Frame interarrival time.
- MOS test value.

Experiments are performed 8 times for each scenario. The collected statistics are processed, and the following statistical parameters are calculated:

- Average packet loss rate at the core router
- Average forwarding delay at the core router
- Standard deviation of average forwarding delay at the core router
- Average number of video frames lost

- Average PSNR value
- Standard deviation of PSNR value
- Average buffer occupancy at client.
- Standard deviation of average buffer occupancy at client
- Average frame interarrival time.
- Standard deviation of frame interarrival time
- MOS rating

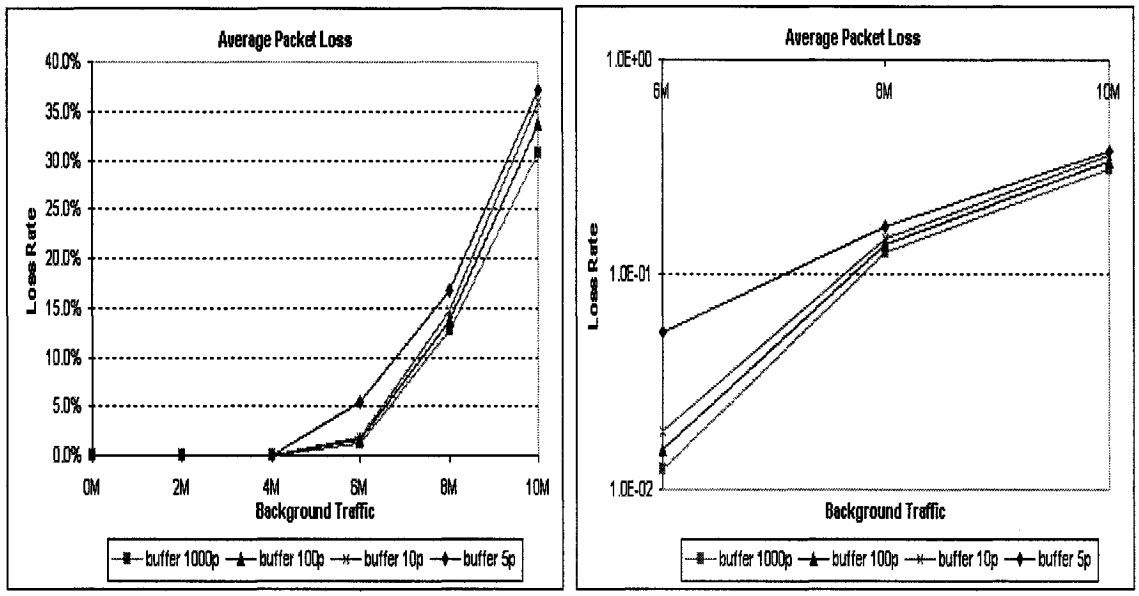
As mentioned earlier, the *ppdump* software (*tcpdump* variant), running within the core router is used to capture the incoming and outgoing packets. In each case, the packet ID is recorded, along with the time stamp of the packet entering the computer from the network interface at the input (incoming packet), or entering the network interface at the output (outgoing packet). Two files are generated in each run, which are exported and processed. Using the data recorded on these two files, we are able to extract the results reported below. In the client side, by capturing the mpeg video packet over the network, we reconstructed the MPEG2 video stream file and used it for calculating PSNR.

5.2.3.1 Performance under CBR traffic

This set of experiments is conducted with the GN Nettest traffic generator producing packets at a constant rate (thus emulating CBR traffic). A best effort traffic discipline is used at the core router. The background traffic is constant bit rate and consists of UDP packets. The volume of competing traffic affects the video traffic considerably. In our testing scenario, for convenience, we use 0Mbps, 2Mbps, 4Mbps, 6Mbps, 8Mbps and 10Mbps as background traffic load. While between zero background traffic loading and 6 Mbps traffic loading, as shown in the figure 5.22, there is no packet loss occurring while packets are lost at 6Mbps because of our measuring scale selection. The packet loss does not start at 6 Mbps, in fact, it starts earlier. The table 5.1 indicates the packet loss as they occur between 5Mbps and 6Mbps. Because there are 800,000 packets transmitted for each case, we can obtain the number of counted error by 800,000 times packet loss rate.

Table 5.1. Packet Loss Rate Versus Traffic Load

Background Traffic (bps)	Packet Loss Rate	Number of Counted Error
5.4M	0.19%	1520
5.5M	0.4%	3200
5.6M	0.5211%	4169
5.7M	0.676%	5408
5.8M	1.268%	10144
5.9M	1.639%	13112

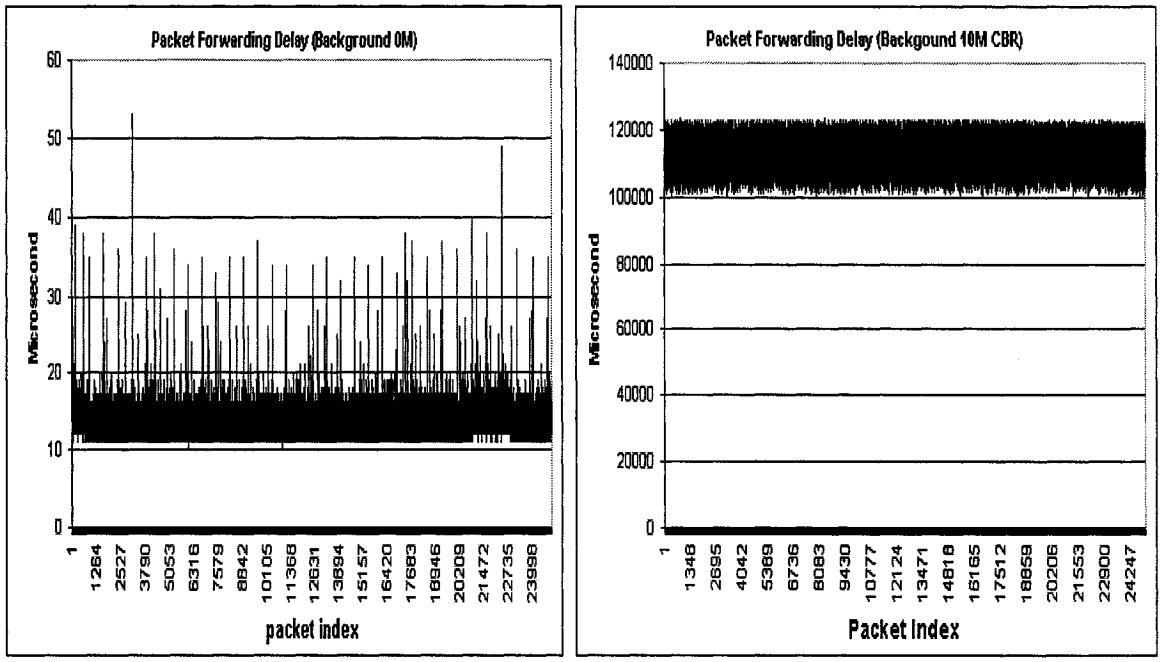


(a) (b)

Figure 5.22 Average Packet Loss Rate experienced by video packets at the core router versus traffic load. (a) linear (b) logarithm scale graphs

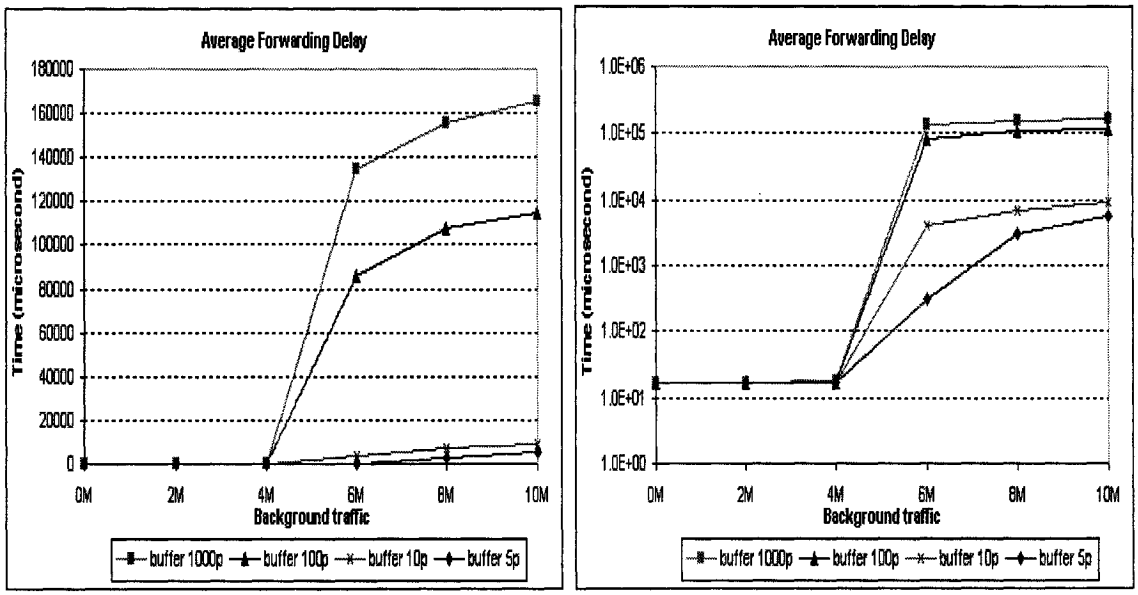
We can see that the packet loss rate changes considerably when background traffic is applied from 5.4Mbps to 10Mbps. The reason for this behavior can be explained as follows. The aggregate traffic (4 Mbps video + 6 Mbps background traffic) produces an average aggregate rate of 10 Mbps, which is equal to the service time of the output link at the core router. At the beginning, the buffers are empty, thus as the packets arrive, they become stored waiting for processing. Due to the increased heavy loading, the queue size grows larger, thus, packets coming later in time are facing higher delays. Figure 5.24

shows the packet forwarding delay in the core router without background traffic and 10Mbps background traffic. At zero background traffic loading almost all packets (with only a few exceptions) experience packet forwarding delay below 40 μsec, with the large majority having delays below 20 μsec, this changes considerably when background traffic is applied. As the background traffic increases, especially over 6 Mbps, all packets experience very high values of forwarding delays in the range of 100 msec to 120 msec, as shown in Figure 5.23. If the buffer size is not long enough, the buffer cannot hold the incoming packets and the incoming packets will be dropped. With the traffic stream reaching volumes higher than the service rate, buffer overflow occurs fast and packets are dropped at large numbers. Figure 5.22 presents this phenomenon: buffer size 5 packets experiences more packet loss than buffer size 1000 packets. Figure 5.24 and 5.25 show us the average packet forwarding delay and its standard deviation experienced in the core router. It is evident that short buffer makes delay lower than long buffer. The delay time increases as the traffic load and buffer length increase. Though video packet losses for buffer sizes 10, 100, 1000 packets are very close, there is a considerable difference in the value of average delay. The value of delay at high loads is more than one order of magnitude higher for BS=100 and 1000 packets as compared to BS=10. As the traffic is close to CBR, what really affects dropping rate of packets is the service rate, since buffer does not provide assistance by buffering and "smoothing" traffic. The system starts losing packets when the traffic load increases to the point it becomes equal or greater to the service rate. Since the main factor in determining losses (in the CBR cases) is the ratio of service rate to traffic load, the losses are very close. However, the delay is a different case. For long buffers, the packets are buffered for longer time, and the delay is higher.



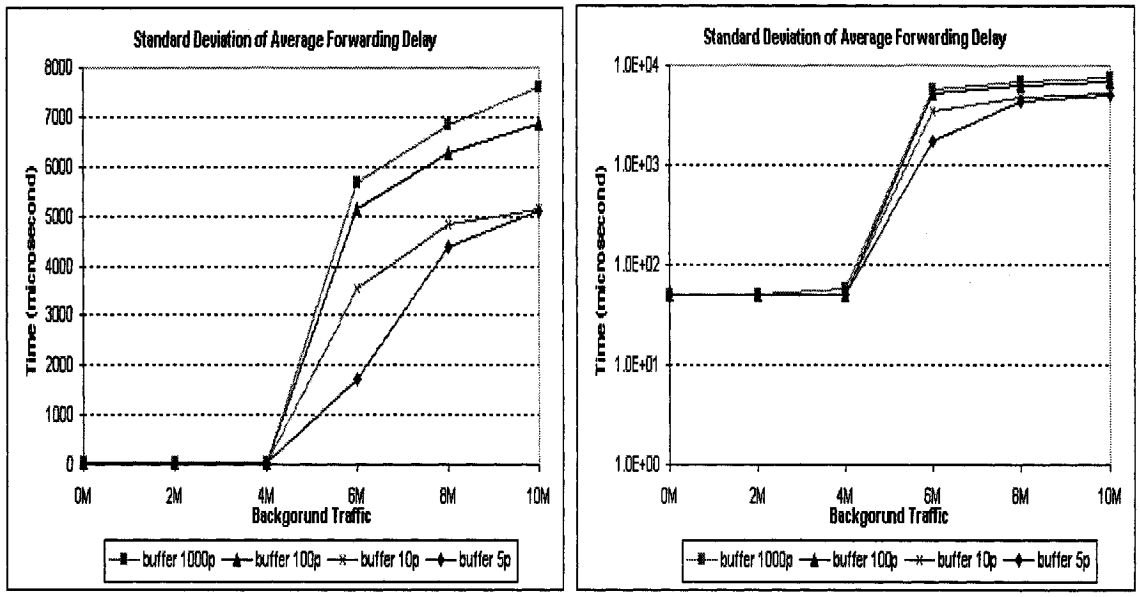
(a) (b)

Figure 5.23 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load (a) 0Mbps and (b) 10Mbps (buffer 100 packets).



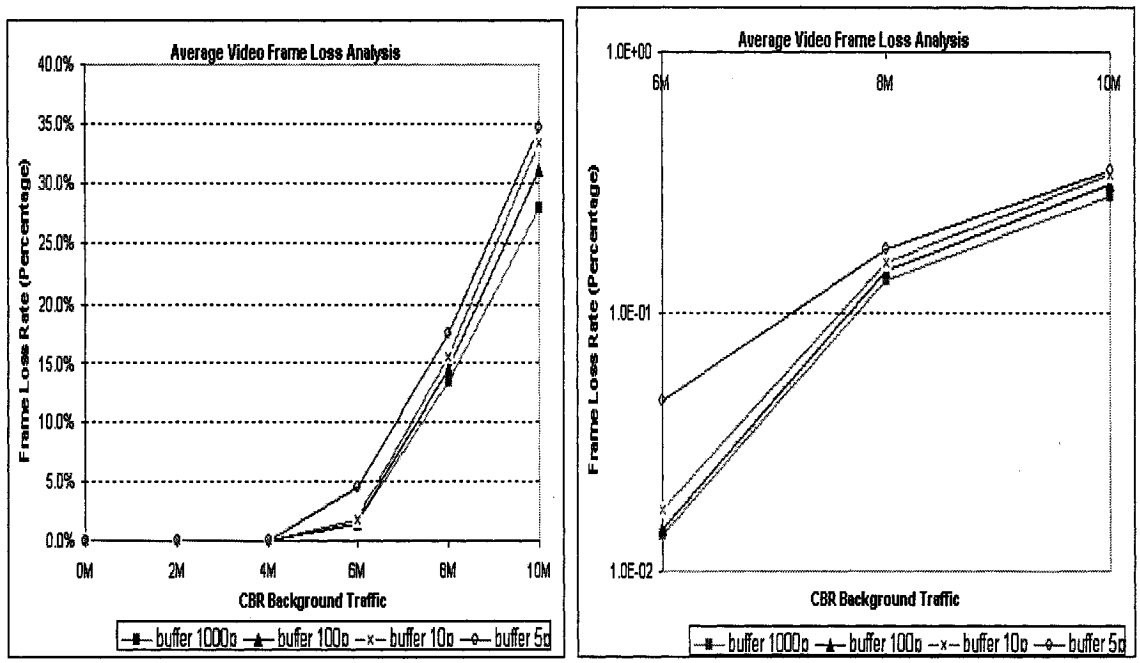
(a) (b)

Figure 5.24 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load (a) linear (b) logarithmic scale graph.



(a) (b)

Figure 5.25 Standard Deviation of Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load (a) linear (b) logarithmic scale graph.



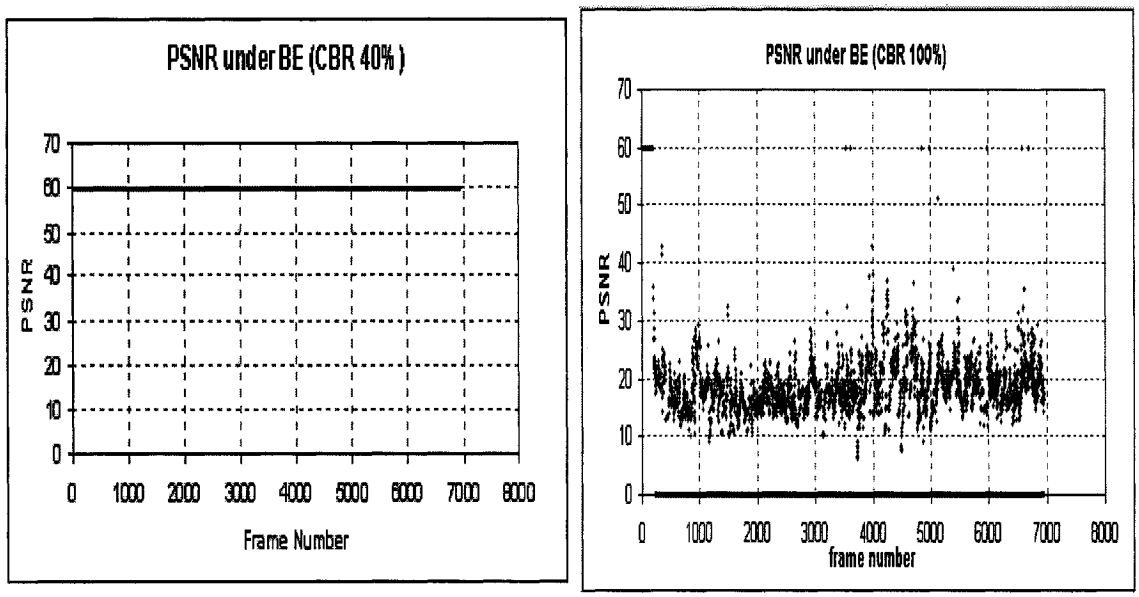
(a) (b)

Figure 5.26 Average Video Frame Loss versus traffic load (a) linear (b) logarithmic scale graph

Fig 5.26 gives the average video frame loss analysis. By comparing Figures 5.22 and 5.26, it is evident that the packet loss rate at the network layer and frame loss rate at mpeg2's application layer are almost identical, this is due to the fact that loss of a frame occurs when the header of a video frame becomes lost. As the header is 188 bytes only, it fits in a single IP packet. IP packets carrying frame headers are subjected to the same impairments and experienced the same probability of loss with the remaining packets, thus the loss of IP packets carrying mpeg2 frame headers is very close to the packet loss rate. This is an important observation as it allows us to project performance parameters of mpeg2's application layer to performance parameters (packet loss rate) of the network layer, thus provides a "lead" for the service provider to monitor the quality. Please note that while damaged frames are "annoying", lost frames are more destructive, since they produce "freezing" of images and temporary loss of the real-time values for the application.

As an important parameter for evaluating the video image, PSNR is used in our test result analysis. It compares the decoded YUV format image with the original YUV format image to calculate the PSNR value. The MPEG2's video traffic entering the video client is passed to the video decoder, where the video file is regenerated. In our case, lost packets are replaced with empty packets prior to the stream being passed to the decoder, thus the effect of the lost packets is reflected on the reconstructed video images. The images of the original and reproduced video files are decoded as YUV pictures and are compared with each other. Thus, we can obtain the PSNR value. The Mpeg2 video traffic average bandwidth is 4Mbps with peak rate of 4.8Mbps and 3.2Mbps, so if the background traffic is under 5Mbps, the total traffic is still under 10Mbps; no traffic congestion occurs at the core router. This means no significant video frame or packet loss. Figure 5.27 shows us that up to 4Mbps the PSNR value is 60db, indicating high quality of image. With the background traffic loading increasing, traffic congestion takes place and video losses occur, degrading the quality. Figure 5.27b gives the PSNR value of individual frames when the background traffic is 10Mbps. It is evident that the PSNR value dramatically degrades, dropping from 60dB to around 20dB for damaged frames, while many frames are lost (indicating 0 dB in the figure). Figure 5.28, 5.29 provide

more detailed view of the PSNR graph by displaying the first 1000 frames and 300 frames.



(a) (b)

Figure 5.27 PSNR graph with background traffic loading (a) 4Mbps and (b) 10Mbps

Figure 5.30 shows the average PSNR and its standard deviation. It is evident that longer buffer size provides better PSNR value than short buffer size. The reason is that long buffer size can hold more packets during the transmission and decreases the probability of video packet and video frame loss rate, thus improving the PSNR performance. From Figure 5.30b, we can see that the standard deviation is very high for 6Mbps background traffic, reading a value larger than 18dB. This is because some frames are on good quality (with values of PSNR in the range of 60dB), while other are on poor quality or lost (0 dB). The PSNR value switches frequently between 60dB and 0dB. The standard deviation gets smaller at 8Mbps and 10 Mbps loading. This is because many frames are lost, getting zero value, and no intact frame (60 db) is received by the client, damaged during the transmission.

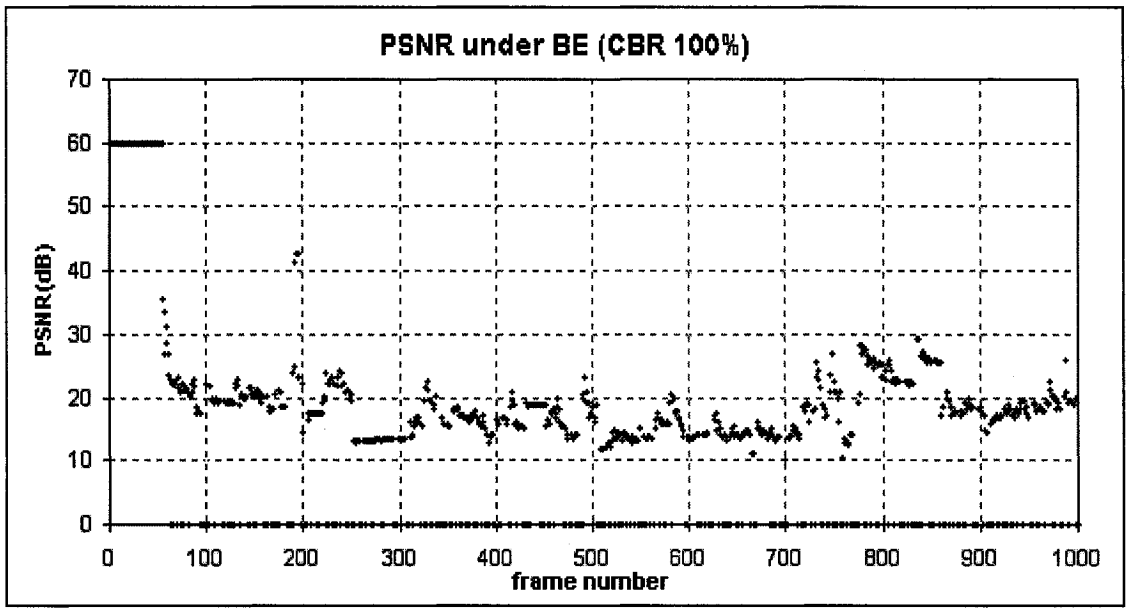


Figure 5.28 PSNR value of the first 1000 frames of the mpeg2 video under 10Mbps traffic loading

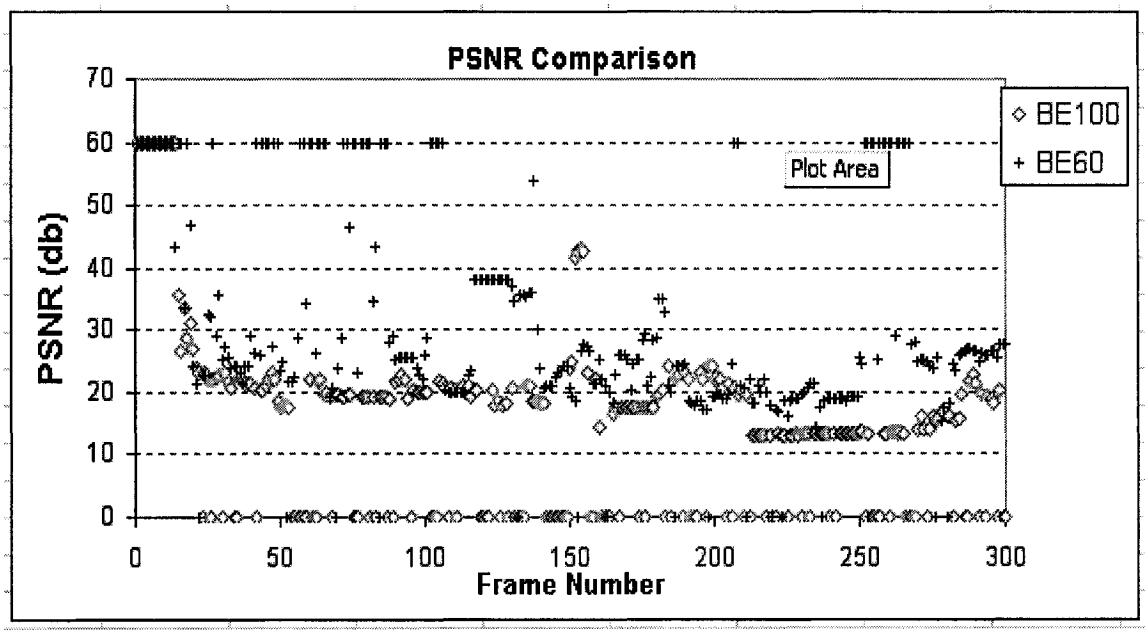
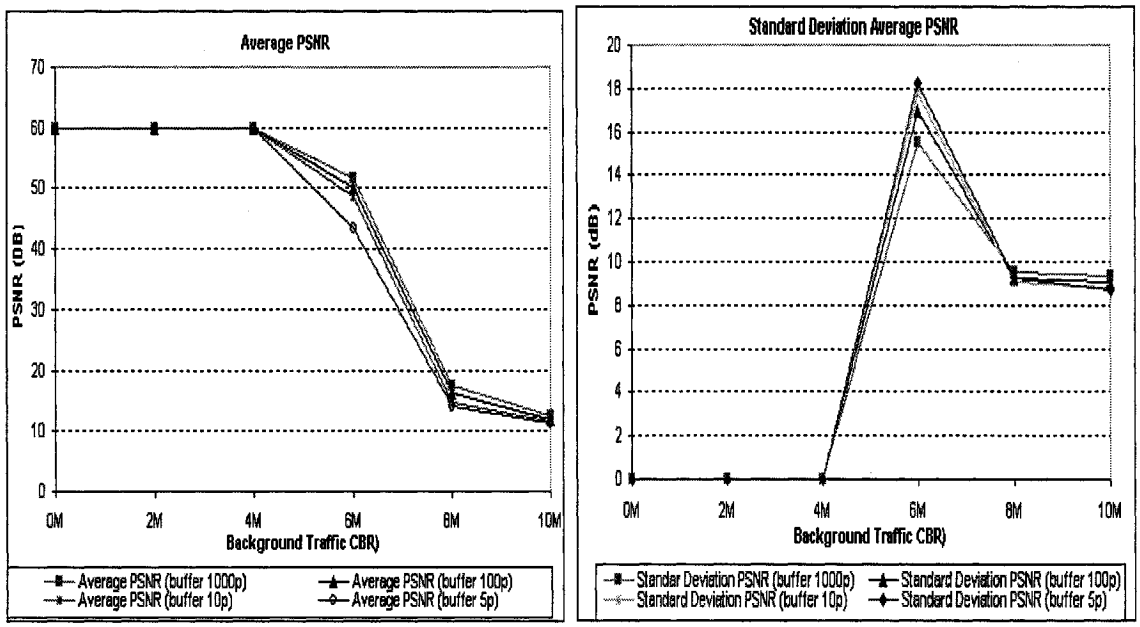


Figure 5.29 PSNR values of first 300 mpeg2 frames under 6Mbps and 10Mbps traffic loading

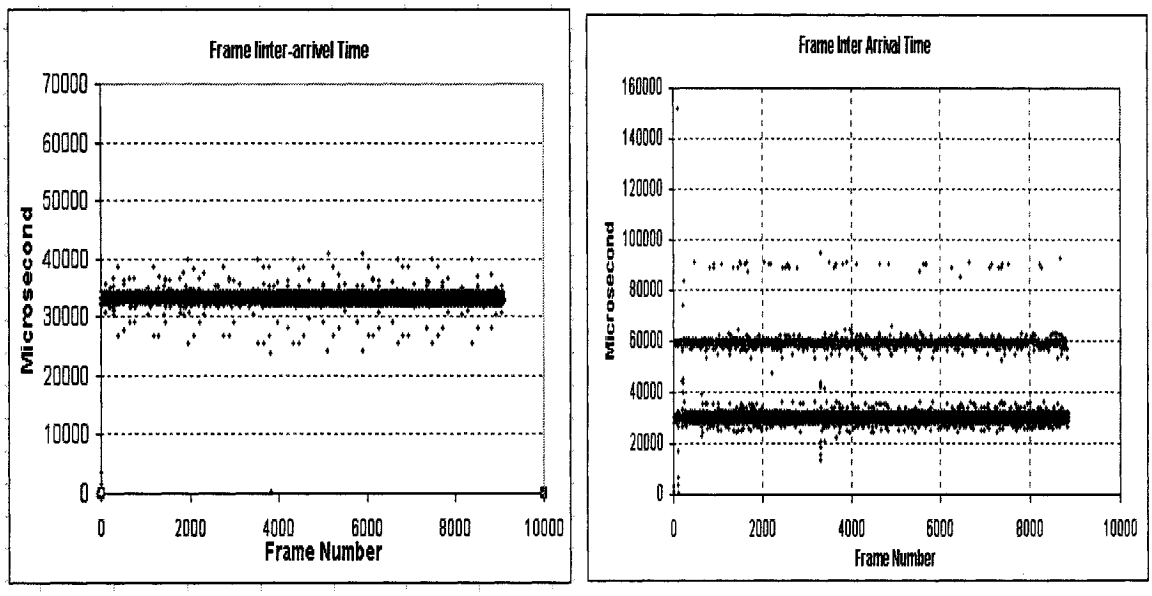


(a) (b)

Figure 5.30 (a) Average PSNR and (b) standard deviation versus background traffic loading

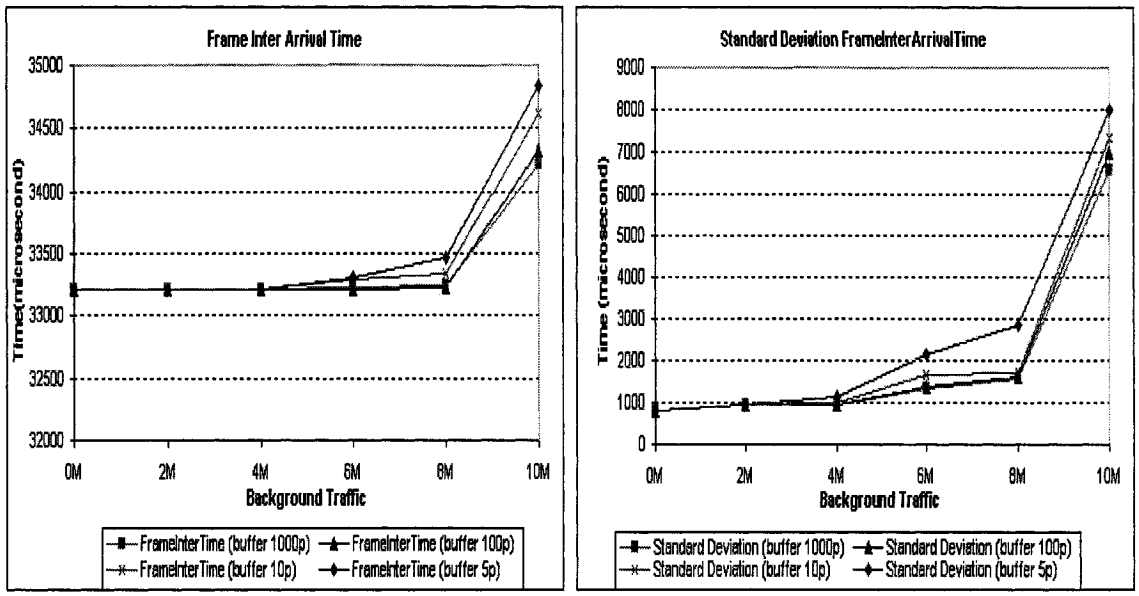
Along with the other parameters related to the quality of images, the frame arrival time at the client side is critical as well. In the ideal situation, frames should be played at the same speed they are generated. However, queuing delays occurring at the network node could delay packets that are needed to reconstruct the frame. This will increase the frame inter-arrival time. The decoder is affected by the jitter of the inter-arrival time of frames. The larger the jitter, the lower the quality of video becomes. The exact effect of the jitter on the video application cannot be evaluated easily. In order to calculate the frame inter-arrival time, it is important to identify the beginning and end of a frame. When the packets are received at the client's side, the sequence number and frame information they carry are identified. Therefore, we can distinguish different frames by identifying two consecutively arriving video packets. In addition, we can obtain the last packet of a frame and the first packet of the consecutive one. The frame inter-arrival time can be calculated by processing these packets. Figure 5.32 shows the frame inter-arrival times experienced by video frames at the client when the background traffic is 0Mbps and 10Mbps. When there is no traffic loading, the average inter-arrival time is around 33.2ms. Under 10Mbps CBR traffic loading, the video frame inter-arrival time shows considerable deviation. This makes the

jitter of frame inter-arrival times larger. Figure 5.32a, 5.32b give the average frame inter-arrival time statistics of all scenarios. With the load increasing and buffer size decreasing, the average frame inter-arrival time is increasing. By the earlier observation, the jitter increase is mostly due to lost frames that also affect the average frame inter-arrival time and standard deviation of average frame inter-arrival time results. As we can see, larger buffers give larger average delay and delay jitter if the deviation from the 33.2msecs was due to queuing delays, then, we would have seen that larger buffer size produce higher jitter around 33.2msecs since larger buffer tend to accumulate packets and produce the variability of queuing delay, in fact, this is not the case here.



(a) (b)

Figure 5.31 Frame Inter-Arrival Times recorded at the video client for (a) 0Mbps and (b) 10Mbps traffic loading



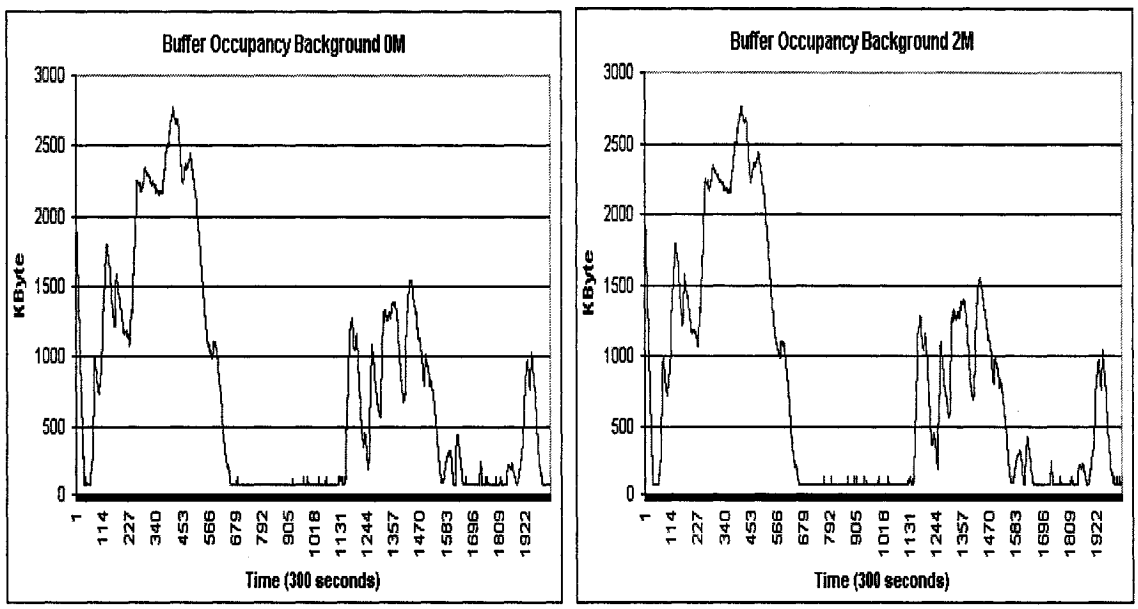
(a) (b)

Figure 5.32 Average Frame Inter-Arrival Time versus traffic load: (a) Average frame inter-arrival time (b) standard deviation

Sustainability of connection refers to the ability of the network to maintain the video connection until its end. In video application, sustainability of the connection is reflected directly by the buffer occupancy at the client side. The decoder decodes the contents in the buffer. If no data in the buffer, the decoder experienced "starve" phenomena and stopped playing video. It is evident that buffer occupancy is an important factor for assessing the video QoS metric. Thus, the amount of video data stored in the client buffer has been monitored during the transmission of the video-clips. The measurements are done at the times when the decoder device requests 64KB of data to be extracted from the buffer.

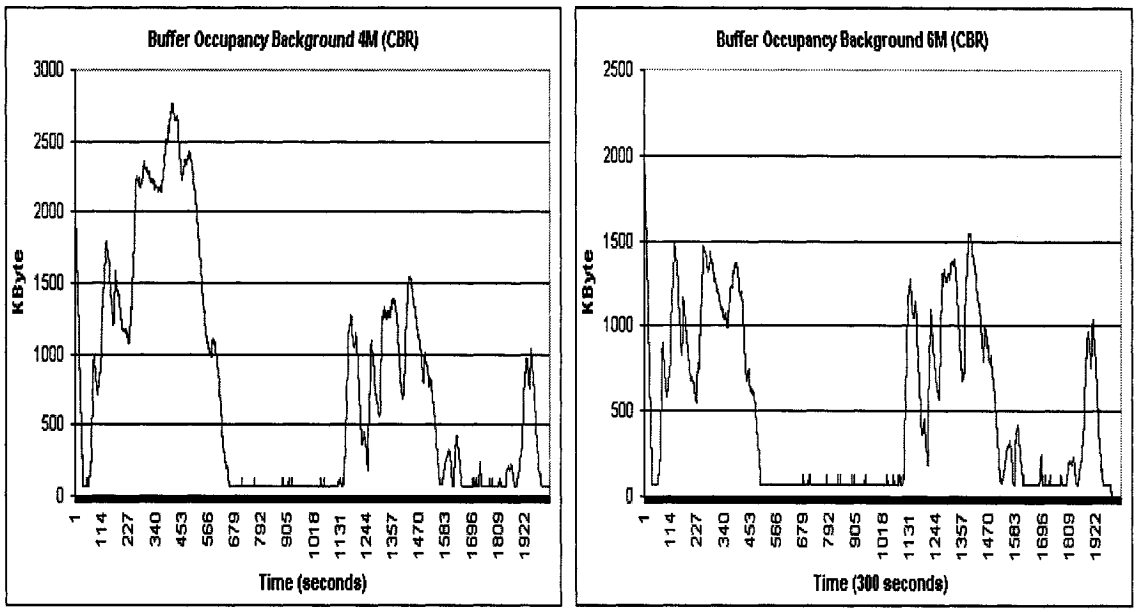
In the following figures, we display the buffer occupancy measured at the client for a best effort network, under different traffic loading conditions. When the network is loaded, the loss of packets due to the network congestion results in a reduction in the amount of video data in the client buffer. There is almost no change on the graph shape for a background load from 0 Mbps up to 4 Mbps. For traffic load of 8 Mbps and 10Mbps, the graph shape is changed considerably due to packet losses.

The curves of Figure 5.33a can be taken as "benchmarks", which we can use for comparing the following results under different traffic loading. Figure 5.33b and 5.34a corresponds to the case where the CBR background traffic of 2 Mbps and 4 Mbps. By inspecting these figures, it is evident that the buffer maintains enough video data and does not enter into the starvation area. Figures 5.34b corresponds to CBR background loading of 6 Mbps. Comparing figures 5.33 and 5.34a, it is evident that the buffer occupancy decreases to lower values considerably. The high value has lowered to 1.5MB from 2MB from the previous case. This indicates that large amount of video packets were dropped in the router due to congestion, never reaching the decoder. As a matter of fact, the quality of video seen on the television screen was very poor and PSNR value very low. As we can see figure 5.34b, the decoder has reached the point of starvation (no video data available for decoding) in front of the end of the observation window. Similar but worse performance was observed under 8 Mbps and 10Mbps traffic loading, as can be seen from figures 5.35a and 5.35b. The size of buffer occupancy is reduced considerably, indicating that most video packets are dropped because of congestion. Again, the buffer enters into the starving state earlier and the time of sustainability of connection becomes shorter. The observed quality of video was intolerable.



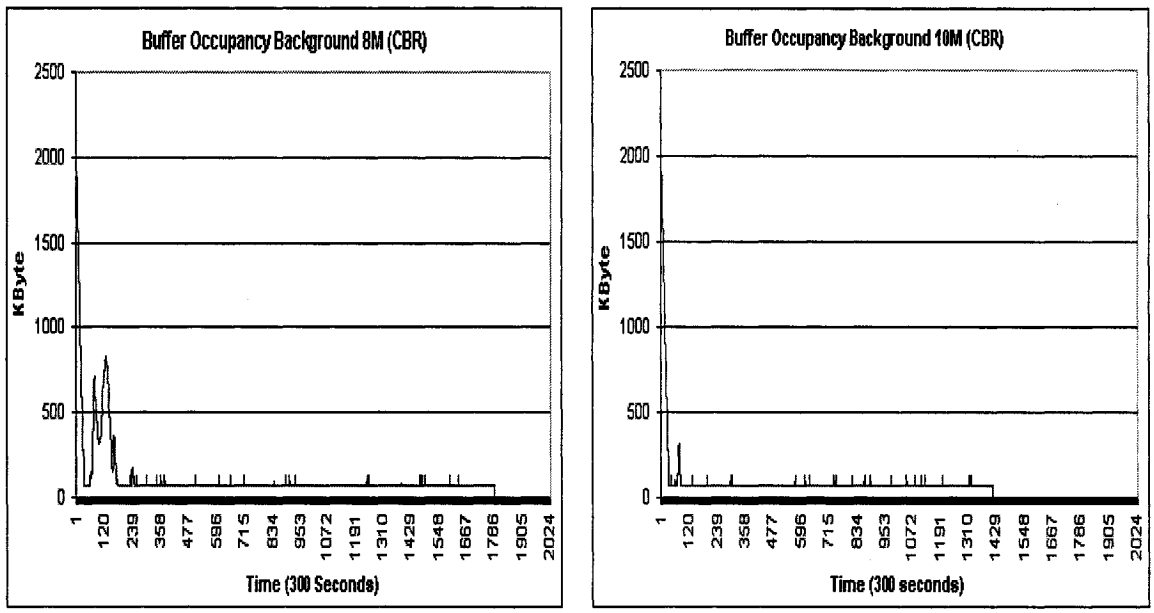
(a) (b)

Figure 5.33 Buffer Occupancy at the decoder with CBR background traffic of: (a) 0Mbps and (b) 2Mbps



(a) (b)

Figure 5.34 Buffer Occupancy at the decoder with CBR background traffic of: (a) 4Mbps and (b) 6Mbps

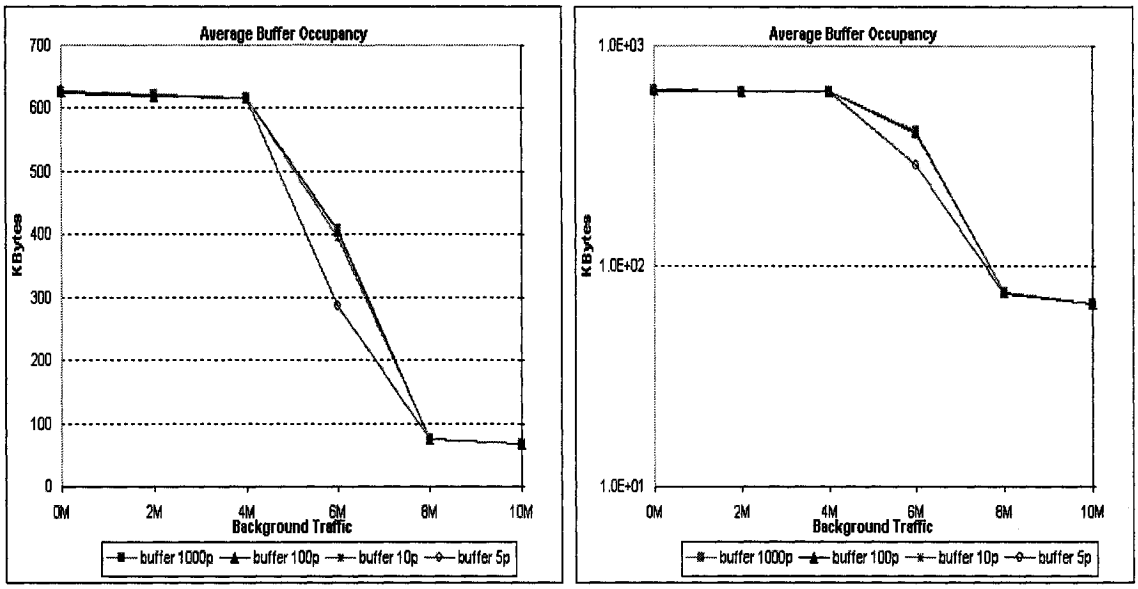


(a) (b)

Figure 5.35 Buffer Occupancy at the decoder with CBR background traffic of: (a) 8Mbps and (b) 10Mbps

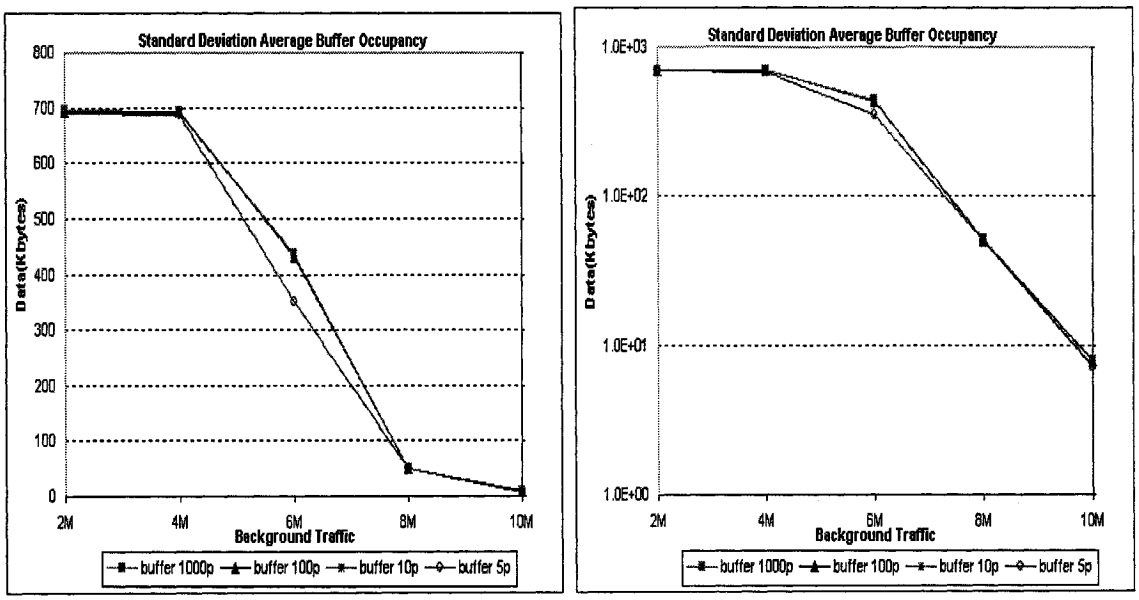
Figure 5.36, 5.37 present the average buffer occupancy and its standard deviation. It is evident that the buffer occupancy decreases largely after the traffic loading over 4Mbps. However, these parameters have not provided convincing proof of packet loss from

8Mbps to 10Mbps because of its average value property. We must combine these parameters with others to evaluate the video quality precisely.



(a) (b)

Figure 5.36 Average Buffer Occupancy versus traffic load (a) linear and (b) Logarithmic scale graph



(a) (b)

Figure 5.37 Standard Deviation of Average Buffer Occupancy versus traffic load (a) linear and (b) Logarithmic scale graph

Figure 5.38 shows the MOS (Mean Opinion Score) graph. The MoS reflects the visual effect that the person perceives (in our test, the opinion of eight persons is sampled for every run). During the high traffic loading of 8 Mbps and 10 Mbps, the quality of video experiencing more blur even frozen images is very poor, intolerable for the viewers. From the curve we can see that its trend is similar to the PSNR trend. Therefore, from the objective view, the MOS index provides another way to evaluate the video quality.

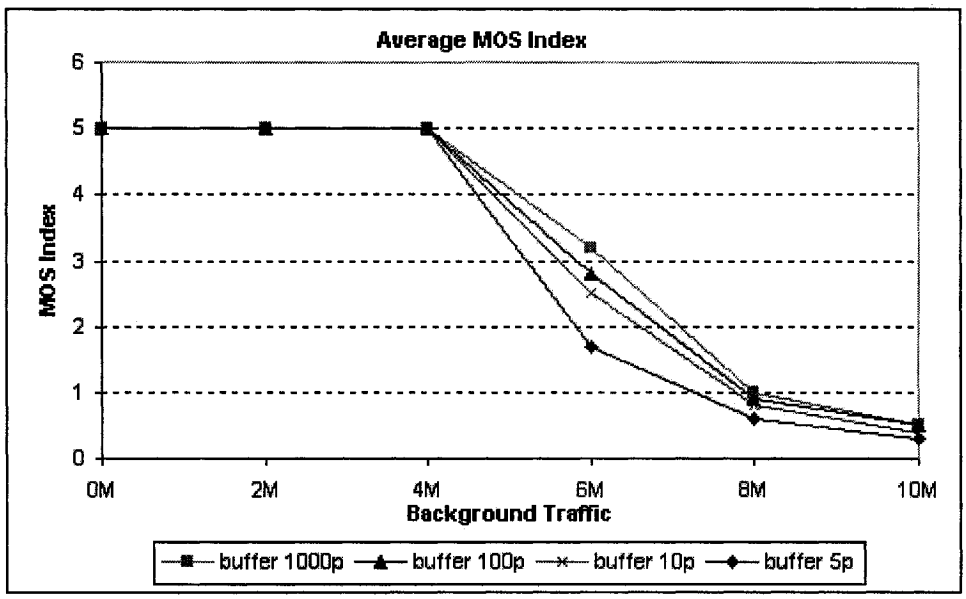


Figure 5.38 Average MOS Index Value

5.2.3.2 Performance under ON-OFF VBR traffic

In this set of experiments, the GN Nettest traffic generator is producing variable ON (active) OFF (inactive pattern) traffic. This generates higher burstiness in the system and allows assessing its impact on the system. The burst traffic parameters are as follows:

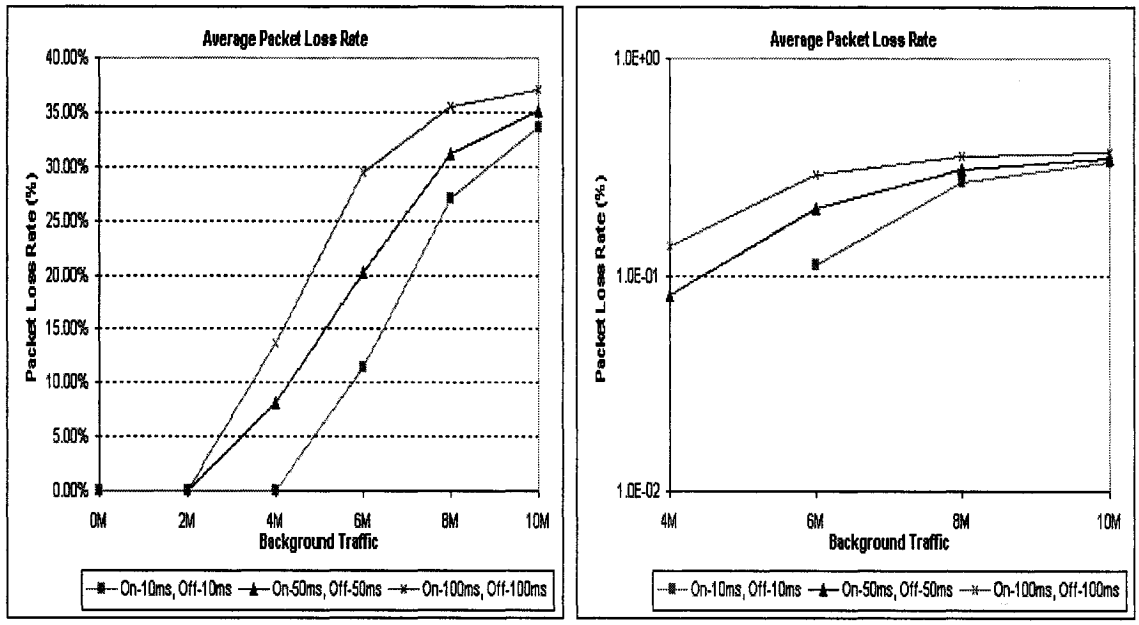
- ON 10ms, OFF 10ms
- ON 50ms, OFF 50ms
- ON 100ms, OFF 100ms

The experiments run 8 times, sending 100,000 packets every time. This gives a total of 800,000 packets for every scenario.

The burst background traffic (VBR) loading generates considerable different impact on Mpeg2 video traffic compared with CBR traffic. We know that under CBR traffic loading, video traffic does not experience packet loss when no congestion occurs. However, in the VBR case, from 5.40 to 5.43, we can see that short buffer size case experiences more packet loss even under low traffic loading. The reason for this phenomenon is that even the average VBR traffic loading is low, its peak still reach the volume that can cause congestion. For short buffer size 5 packets and 10 packets, the packet loss rate under 10ms-10ms VBR traffic is better than those under 50ms-50ms and 100ms-100ms VBR traffic, because short buffer cannot hold the long burst and is forced to drop the incoming video packets. However, for long buffer size 100 packets and 1000 packets, the packet loss rate under 10ms-10ms VBR traffic is worse than those under 50ms-50ms and 100ms-100ms VBR traffic. It is evident that long buffer can process long burst better than short buffer. What we need notice that in figure 5.40a, the logarithm scale graph, because of the zero loss rate for 4 Mbps 10ms-10ms VB, its logarithm value cannot be calculated (infinity), thus the curve for 10ms-10ms VBR is terminated at 6Mbps. we will see similar scenario in following figures.

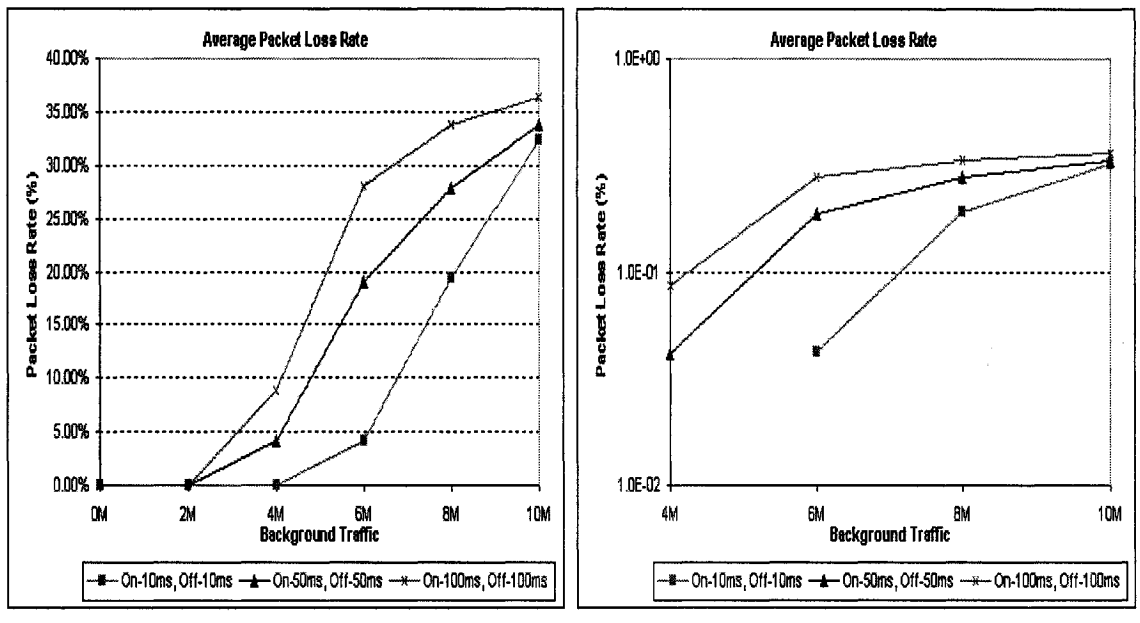
From Fig 5.44 to Fig 5.51, the curves show the average forwarding delay and standard deviation of average forwarding delay. Because of VBR's high volume traffic of ON state, the forwarding delay increases even in low traffic loading (4Mbps etc). It is evident that short buffer generates small forwarding delay because of less queuing occurring within the

buffer. When we compare the delay between CBR and VBR background traffic, we find that the forwarding delay under CBR is higher than VBR under high background traffic loading and smaller than VBR under low background traffic loading. The reason is that VBR traffic loading 4 Mbps means that the traffic peak volume reaches 8 Mbps, thus traffic congestion occurs and makes the forwarding delay to increase. Therefore, compared to CBR, VBR increases the forwarding delay when background traffic is lower than 6Mbps. When the average background traffic increases to 8Mbps or above, the system becomes highly congested. In this situation, CBR traffic always keeps the system loaded. However for VBR, because of its ON-OFF property, the system still has time to process the output traffic without input traffic interference. It is evident that in the VBR case, the system has more time to handle the video packets, thus reducing the forwarding delay of video packets as compared to CBR traffic. More OFF state time means more system time is used for processing packet forwarding. We also noticed that the forwarding delay under VBR 50ms-50ms and 100ms-100ms is lower than that under VBR 10ms-10ms under high background traffic, as well as packet loss rate. It is because 10ms-10ms VBR is short burst traffic and ON-OFF state changes very fast. With the high background traffic loading, the buffer keeps full and cannot be cleared even under OFF state because system cannot react the rapid traffic state change in time. In fact, the short burst traffic can be seen as sort of "CBR" traffic. From above analysis, we find that the response of system to CBR and 10ms-10ms VBR is similar. When the low background traffic loading is applied, the packet forwarding delay under 50ms-50ms VBR and 100ms-100ms VBR is higher than that of 10ms-10ms VBR (e.g. 4 Mbps), especially for short buffer size 5 and 10 packets. At this moment, the packet loss occurs and buffer becomes full for 50ms-50ms and 100ms-100ms VBR, not for 10ms-10ms VBR. It is evident that forwarding delay under 10ms-10ms VBR is smaller than 50ms-50ms and 100ms-100ms and packet loss rate under 10ms-10ms VBR is smaller as well. In addition, because of traffic burst property, this means that some video packets are processed under heavy system load and some under light system load. It is evident that the forwarding delay under 50ms-50ms and 100ms-100ms VBR experiences more deviation than 10ms-10ms VBR, as shown in figures.



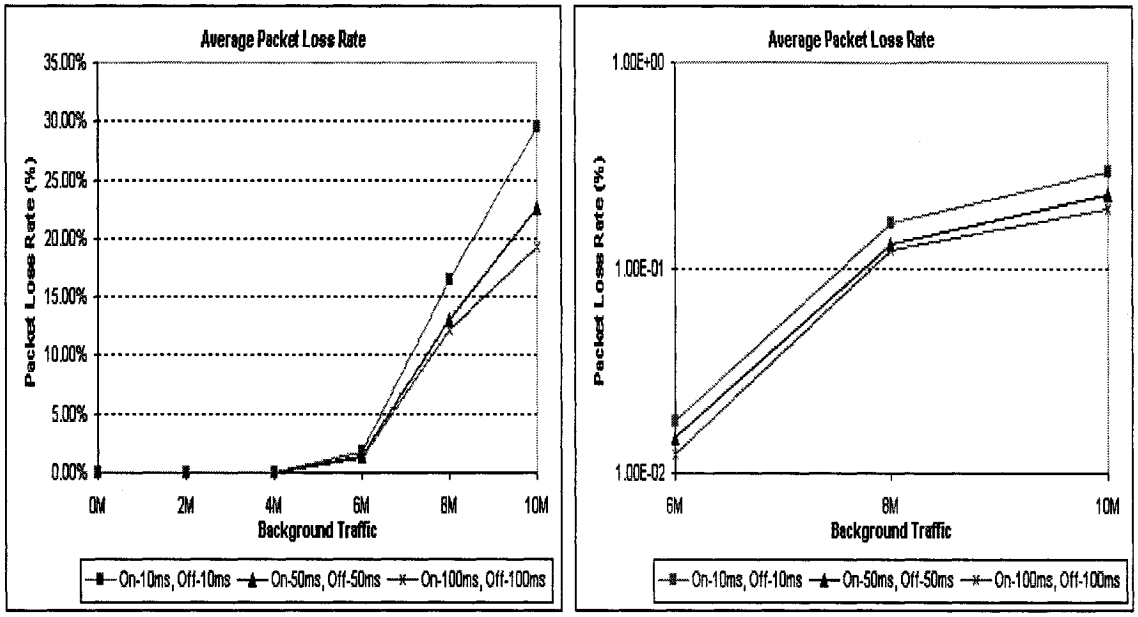
(a) (b)

Figure 5.40 Average Packet Loss Rate experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithm scale graphs (buffer size 5 packets)



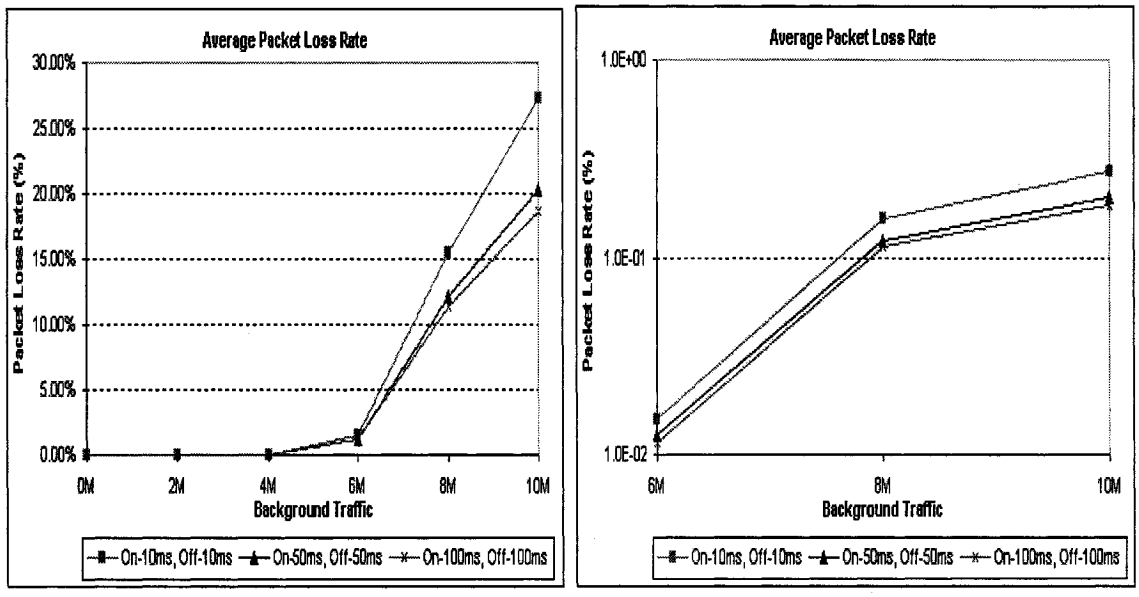
(a) (b)

Figure 5.41 Average Packet Loss Rate experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithm scale graphs (buffer size 10 packets)



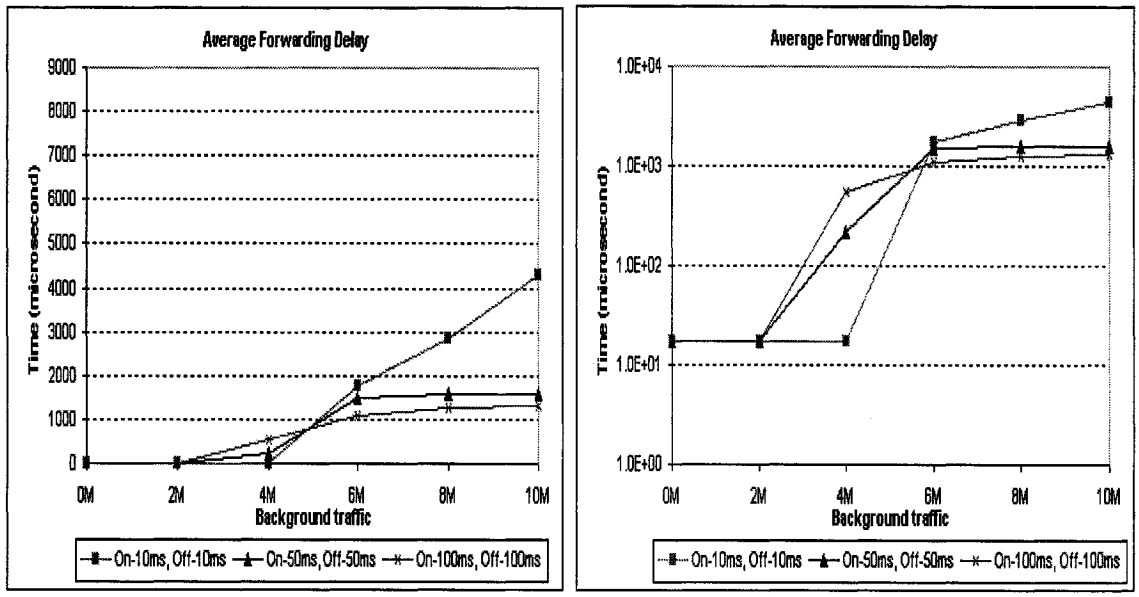
(a) (b)

Figure 5.42 Average Packet Loss Rate experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithm scale graphs (buffer size 100 packets)



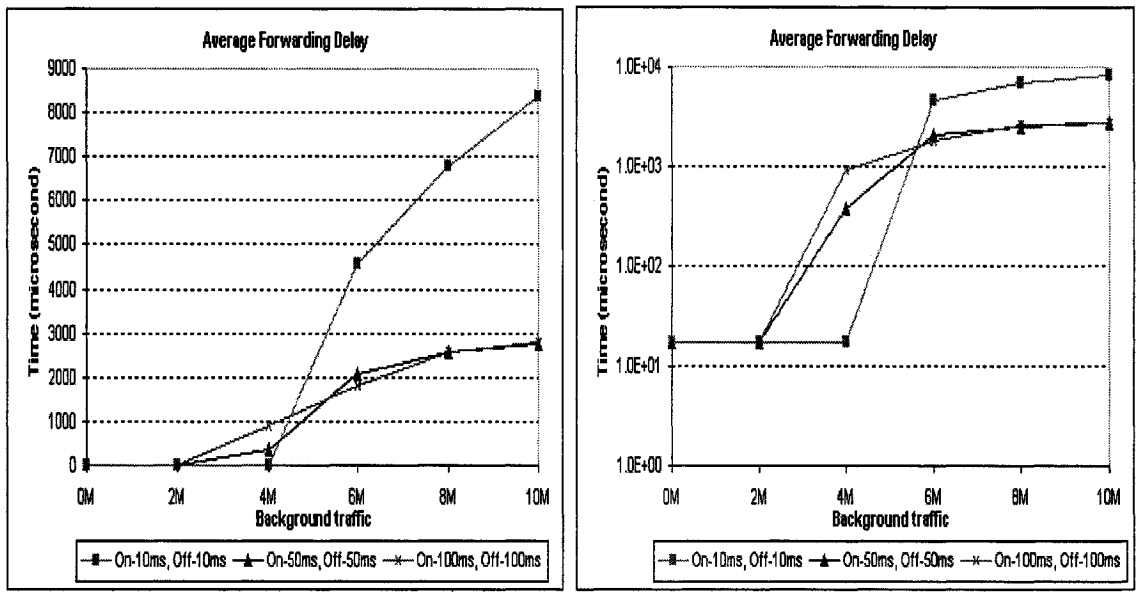
(a) (b)

Figure 5.43 Average Packet Loss Rate experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithm scale graphs (buffer size 1000 packets)



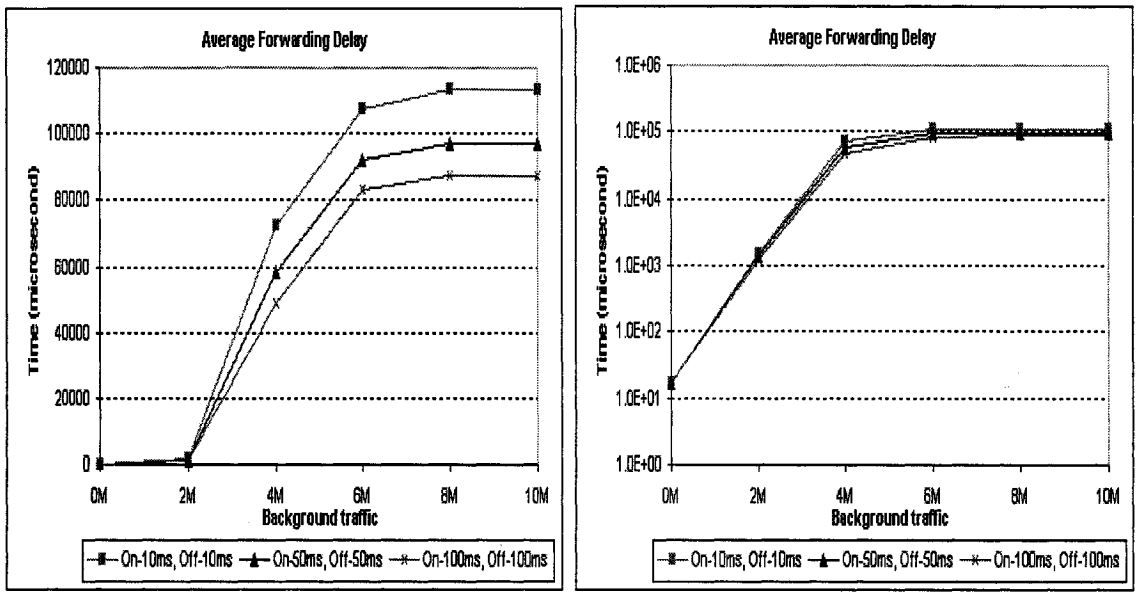
(a) (b)

Figure 5.44 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (Buffer size 5 Packets)



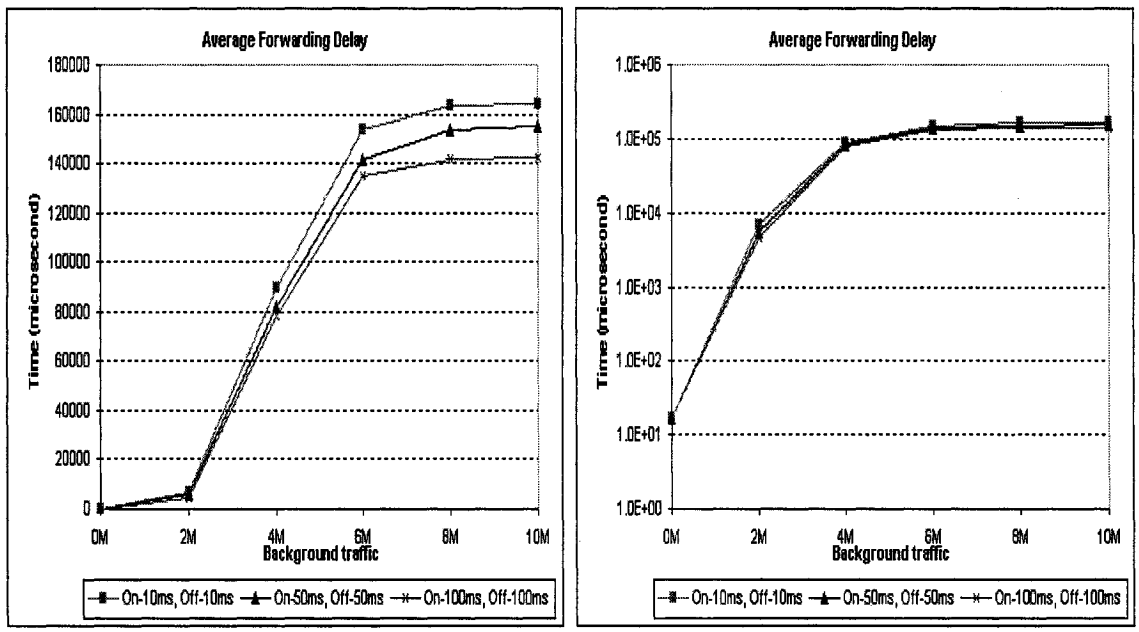
(a) (b)

Figure 5.45 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (buffer size 10 Packets)



(a) (b)

Figure 5.46 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (buffer size 100 Packets)



(a) (b)

Figure 5.47 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (buffer size 1000 Packets)

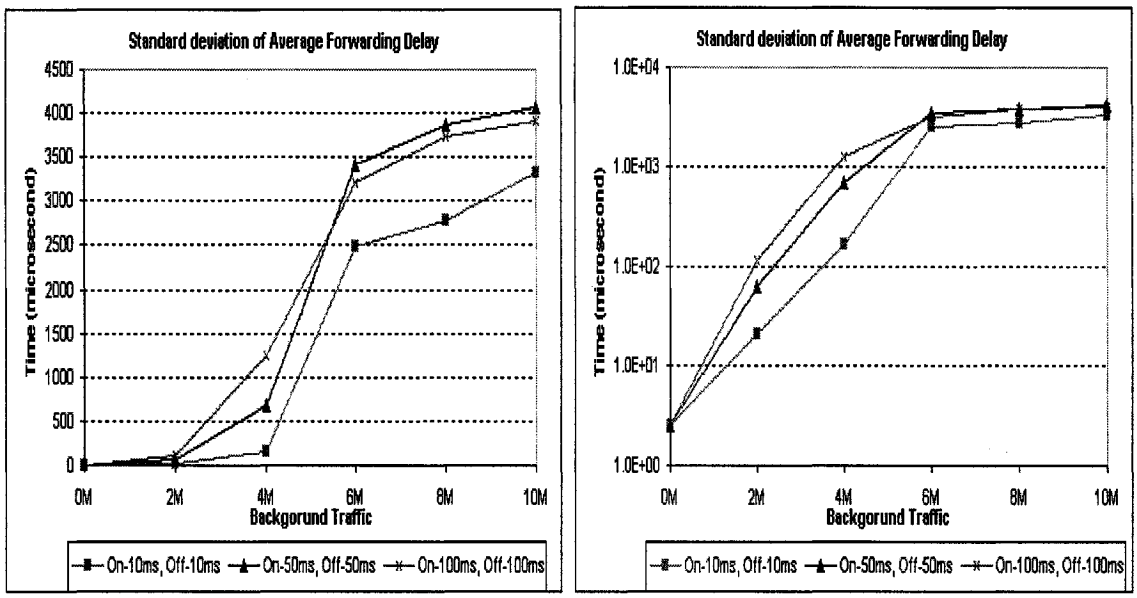
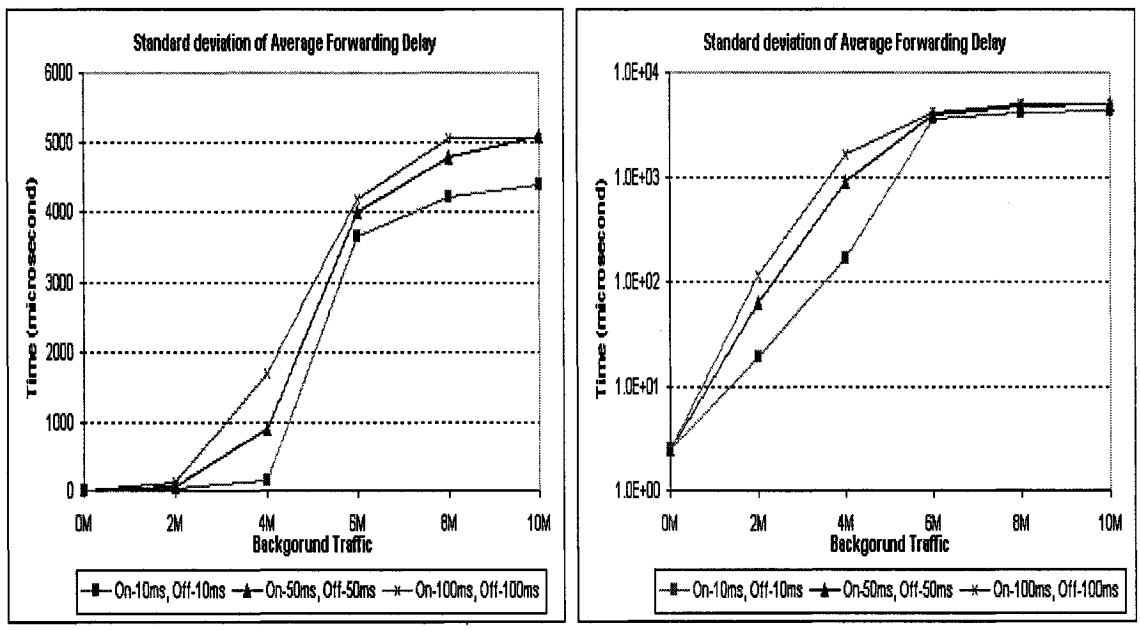
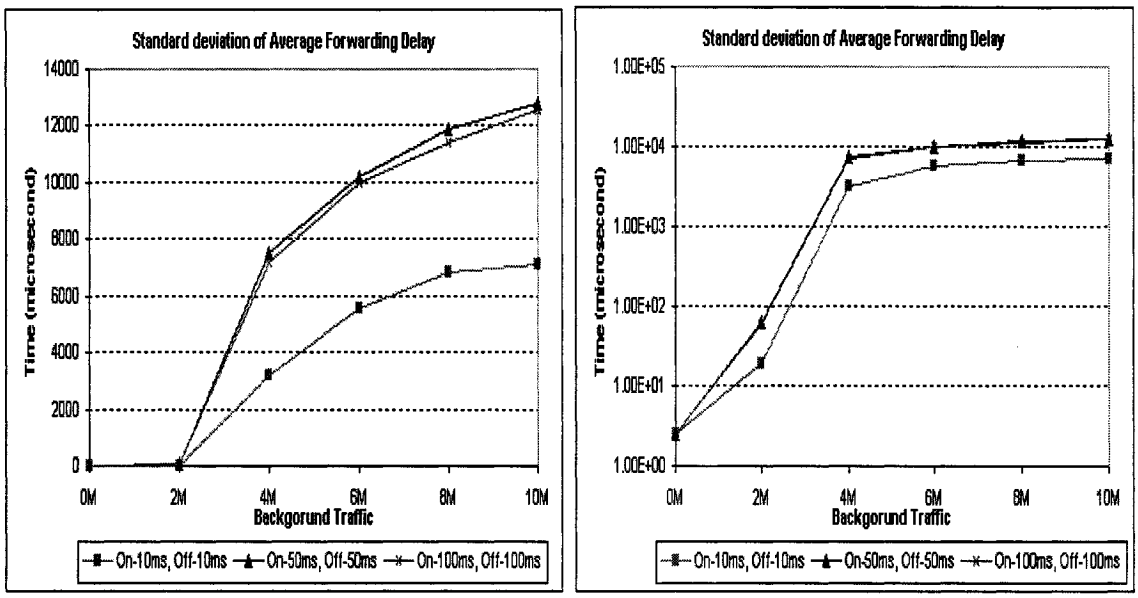


Figure 5.48 Standard Deviation of Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (buffer size 5 packets)



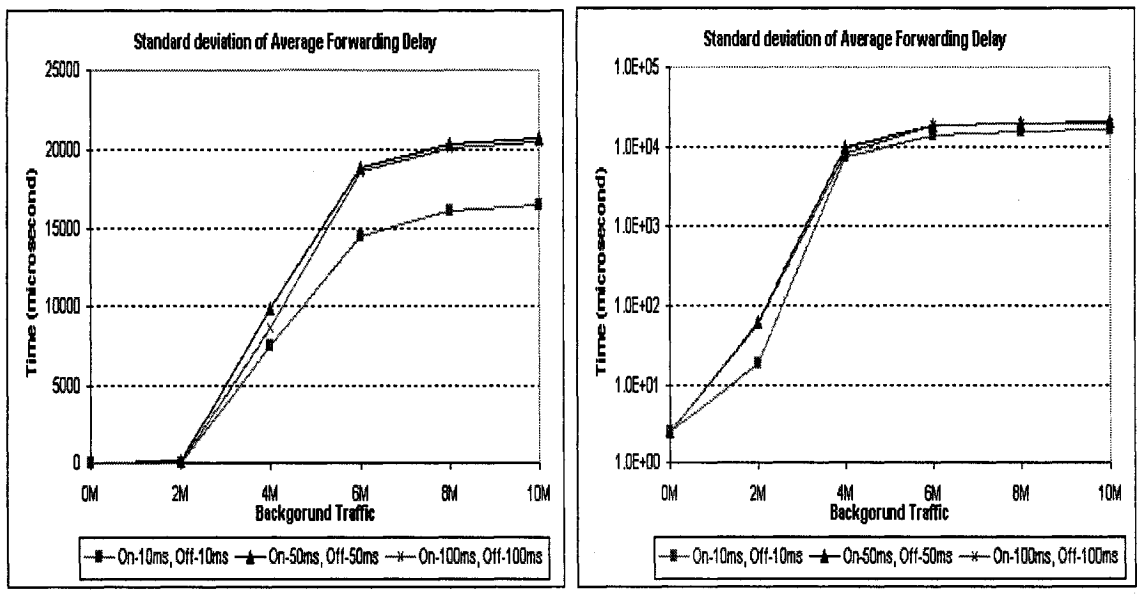
(a) (b)

Figure 5.49 Standard Deviation of Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (buffer size 10 packets)



(a) (b)

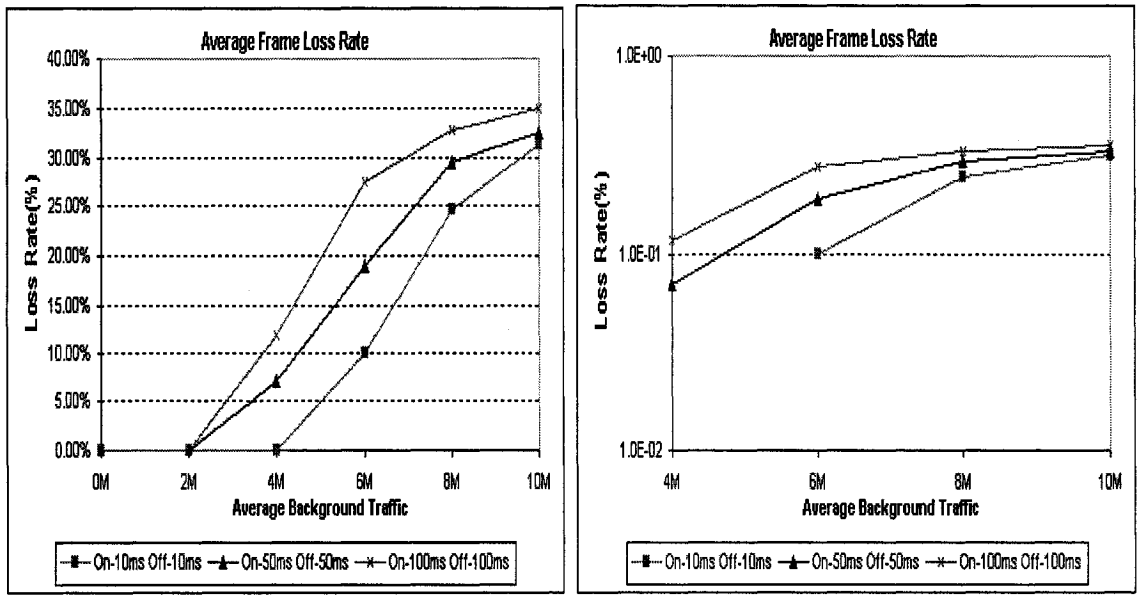
Figure 5.50 Standard Deviation of Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graphs. (buffer size 100 packets)



(a) (b)

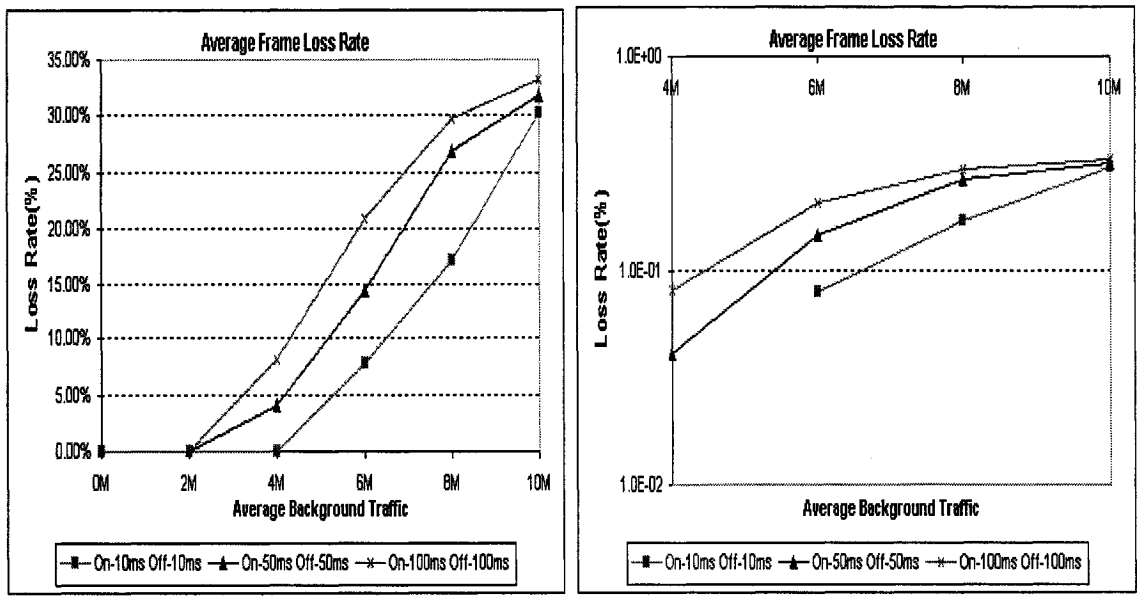
Figure 5.51 Standard Deviation of Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load. (a) linear and (b) logarithmic scale graph. (buffer size 1000 packets)

Figures 5.53 to 5.56 present the frame loss rate and its logarithmic scale graphs for different buffer size under VBR traffic loading. For small buffer sizes (5 packets, 10 packets), the 10ms-10ms VBR traffic has better performance than the 50ms-50ms and the 100ms-100ms in terms of frame loss rate, the trend is similar to that of packet loss rate because frame is reassembled by packets and the video frame header content is encapsulated in the packet load. This is due to the fact that loss of a frame occurs when the header of a video frame becomes lost. As the header is 188 bytes only, it fits in a single IP packet. IP packets carrying frame headers are subjected to the same impairments and experienced the same probability of loss with the remaining packets, thus the loss of IP packets carrying mpeg2 frame headers is very close to the packet loss rate. This is an important observation as it allows us to project performance parameters of mpeg2's application layer to performance parameters (packet loss rate) of the network layer, thus provides a "lead" for the service provider to monitor the quality. Please note that while damaged frames are "annoying", lost frames are more destructive, since they produce "freezing" of images and temporary loss of the real-time values for the application. However, for long buffer size case (100 packets, 1000 packets), 10ms-10ms VBR performance is worse than 50ms-50ms and 100ms-100ms in terms of frame loss rate, this is similar to packet loss rate that we explained earlier.



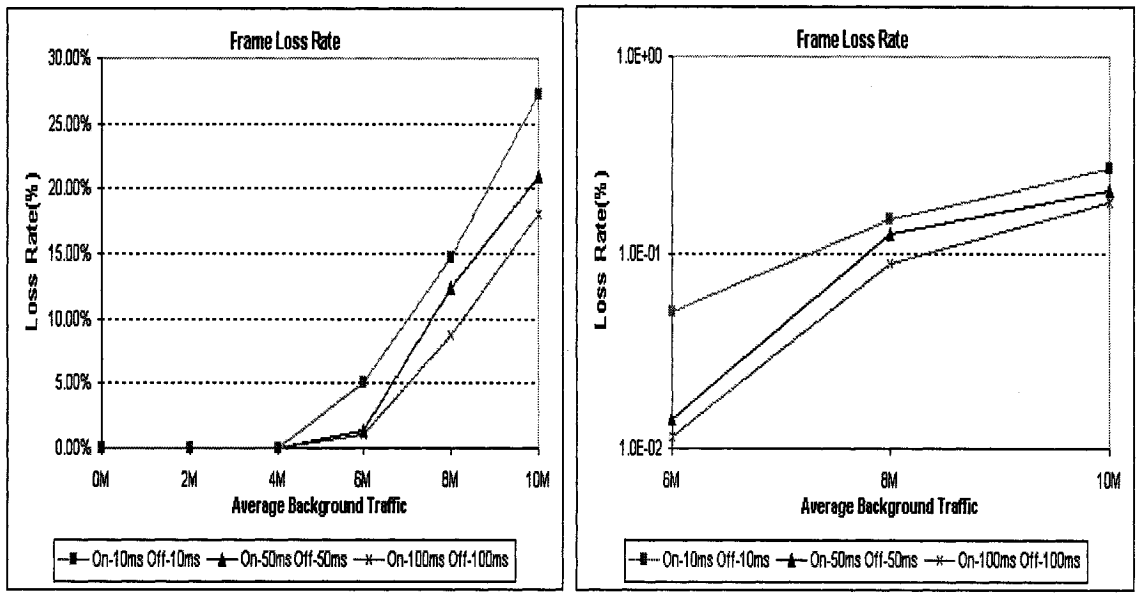
(a) (b)

Figure 5.53 Average Video Frame Loss versus traffic load: (a) linear (b) logarithmic scale graphs. (buffer size 5 packets)



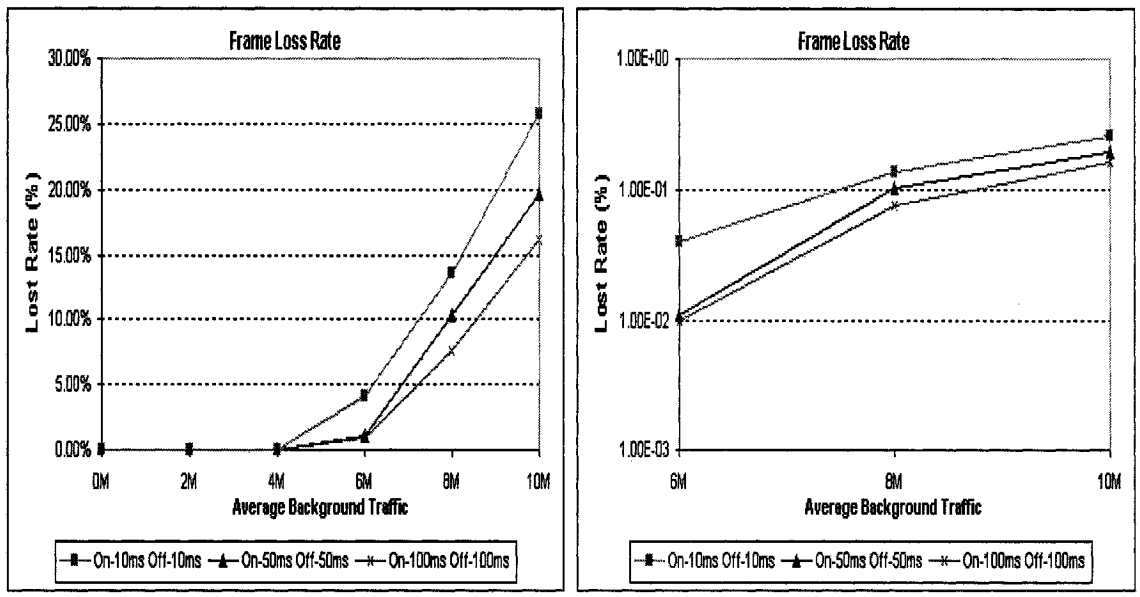
(a) (b)

Figure 5.54 Average Video Frame Loss versus traffic load (a) linear and (b) logarithmic scale graphs. (buffer size 10 packets)



(a) (b)

Figure 5.55 Average Video Frame Loss versus traffic load (a) linear and (b) logarithmic scale graphs. (buffer size 100 packets)



(a) (b)

Figure 5.56 Average Video Frame Loss versus traffic load (a) linear (b) logarithmic scale graphs. (buffer size 1000 packets)

When comparing with the CBR scenario, we find that VBR traffic considerably affects the performance of video (packet and frame loss) even under low traffic loading, where no packet loss for CBR (especially for small buffer size).

The buffer size and background VBR traffic affects PSNR considerably. Small buffer size produces worse PSNR values, as shown in figures 5.57, 5.58. For buffer size of 100 packets, the PSNR fluctuates between 10dB and 30dB (0 dB not included). For buffer size of 10 packets, the PSNR changes between 8dB and 20dB. The reason is long buffer holds more packets and the client gets more visual information to play. The curves on figure 5.58a and 5.59a show that, for small buffer sizes, packets begin to be dropped at 4Mbps and the PSNR drops to 23.6dB (5 packets buffer size) and 32.7dB (10packets buffer size) when 50ms-50ms and 100ms-100ms background traffic loading are applied. However, for long buffer sizes, there is no packet loss until 6Mbps traffic loading, as shown in figure 5.60a and 5.61a, where PSNR maintains at 60dB. The PSNR standard deviation graphs in figures 5.59b, 5.60b, 5.61b and 5.62b indicate that the PSNR value fluctuates in a rather large range when On-Off VBR traffic is applied. For small buffer size, the large variation happens at 4Mbps. This is because some frames are of good quality while other of very poor. For traffic loading over 6Mbps and higher, most frames are of bad quality, thus the variation begins to decrease. For the long buffer size, big variation occurs at 8 Mbps.

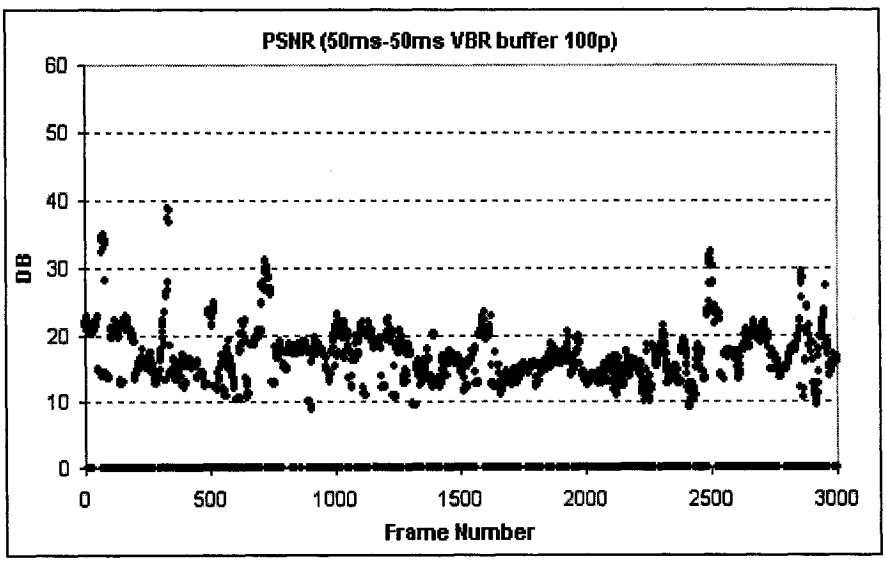


Figure 5.57 PSNR graph of individual frames with VBR background traffic loading (50ms-50ms) and buffer size of 100 packets

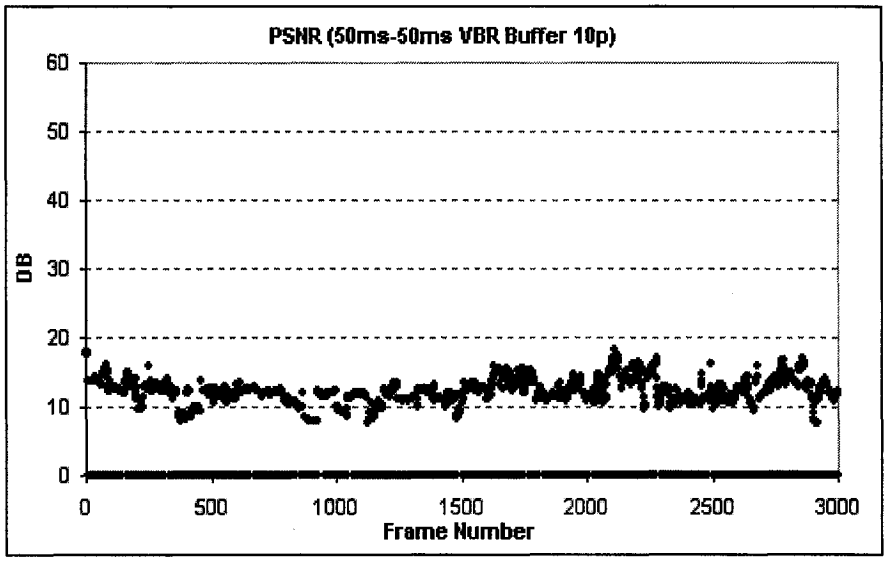
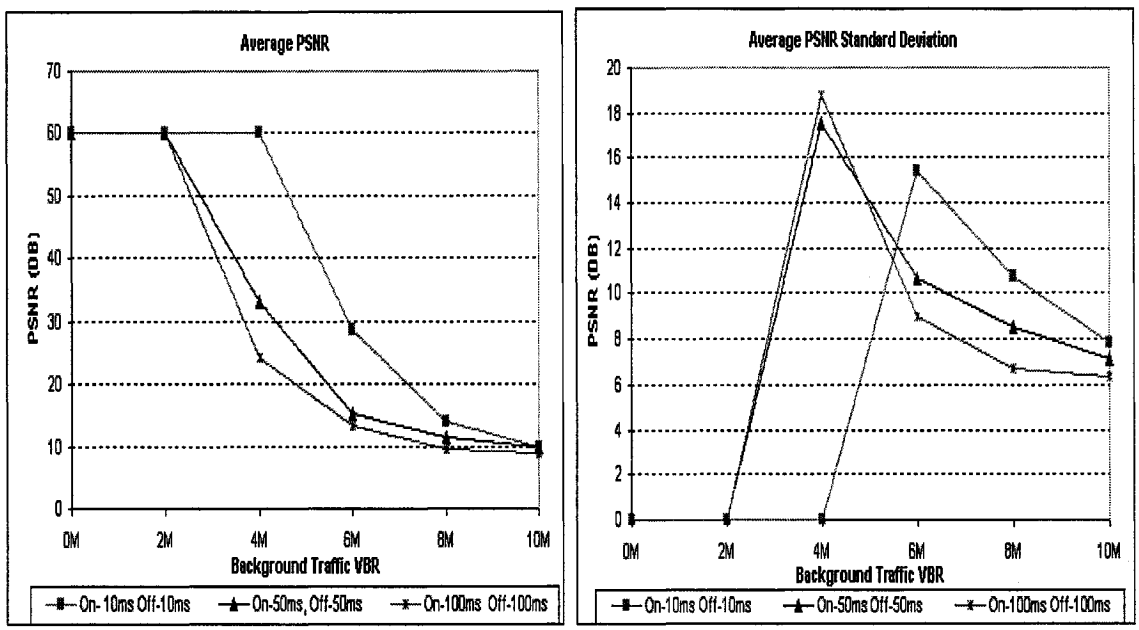
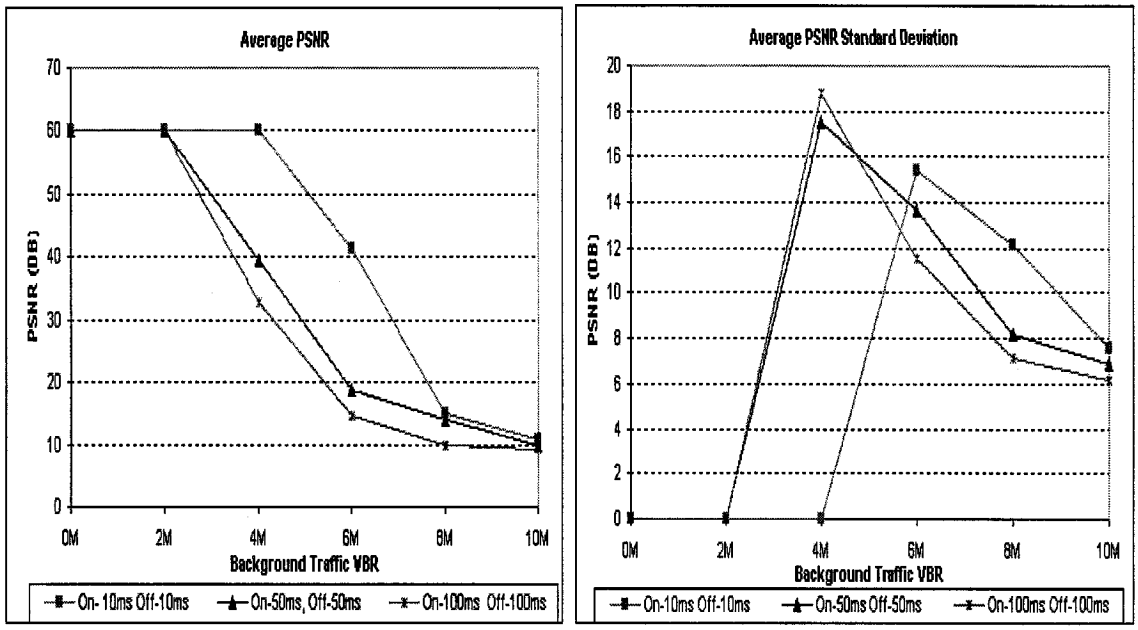


Figure 5.58 PSNR graph of individual frames with VBR background traffic loading (50ms-50ms) and buffer size of 10 packets



(a) (b)

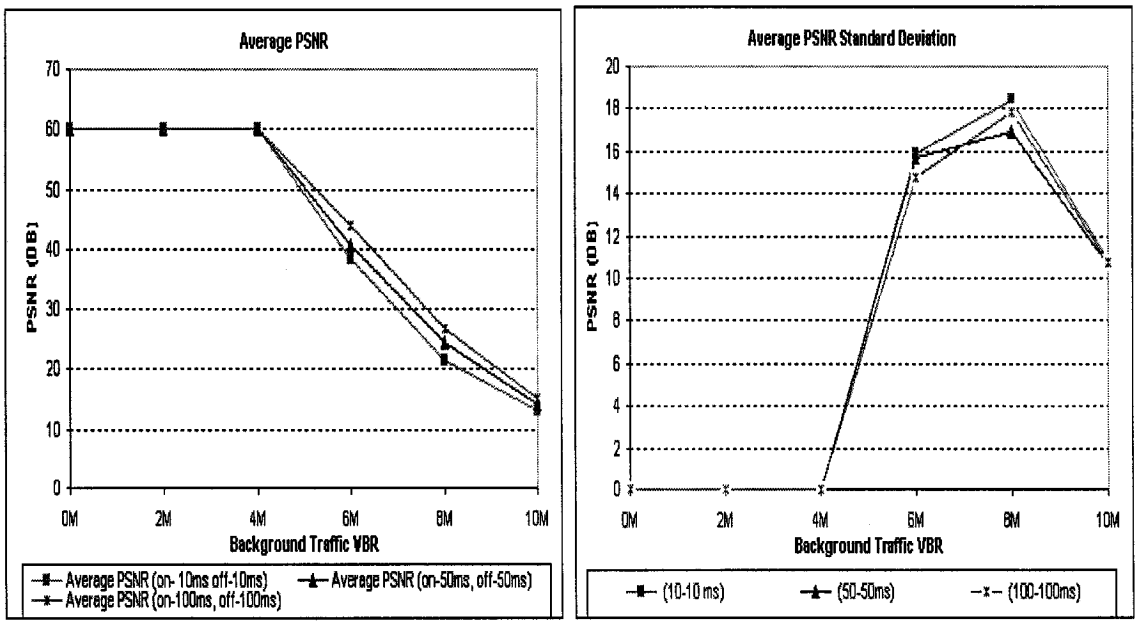
Figure 5.59 (a) Average PSNR and (b) standard deviation versus background traffic loading (5 packets buffer size)



(a) (b)

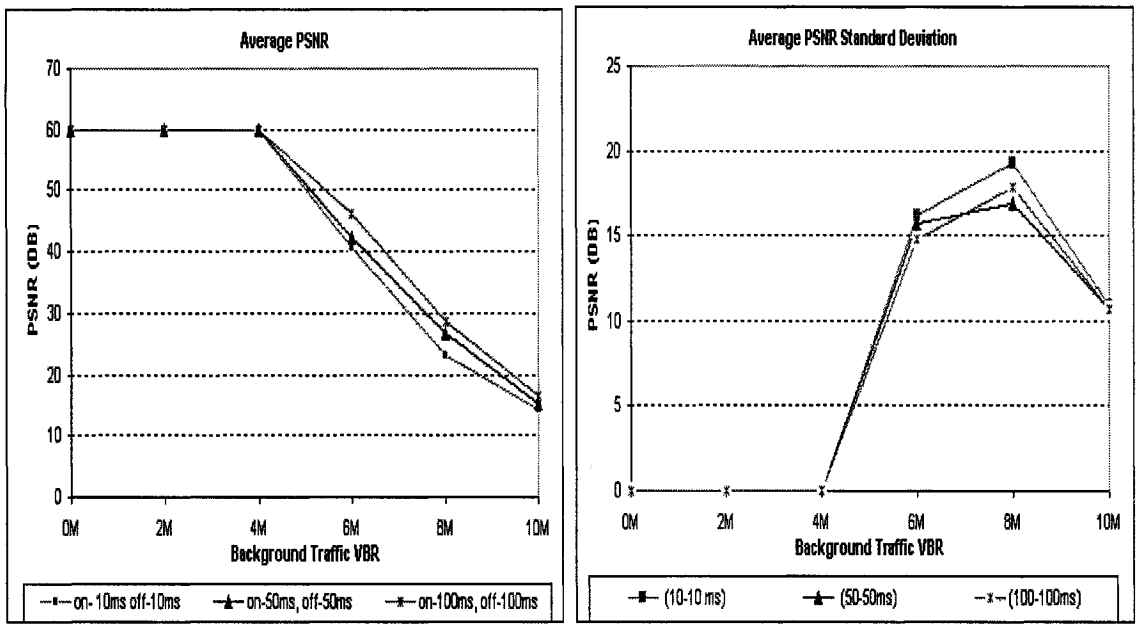
Figure 5.60 (a) Average PSNR and (b) standard deviation versus background traffic loading (10 packets buffer size)

In all cases, the "peak" in standard deviation appears when going from very good performance to experiencing substantial losses. This is consequent, since these points correspond to the packet losses.



(a) (b)

Figure 5.61 (a) Average PSNR and (b) standard deviation versus background traffic loading (100 packets buffer size)



(a) (b)

Figure 5.62 (a) Average PSNR and (b) standard deviation versus background traffic loading (1000 packets buffer size)

Frame inter-arrival time experiences more fluctuations when VBR traffic loading is applied. The curve shown in Figure 5.63 indicates the inter-arrival time experienced by video when background traffic loading is not applied, it can be viewed as a "benchmark case". It is evident that most of the frame inter-arrival times are closed to 33 ms, thus the video client can decode the video frame at regular speed of 30 frames/second. Because we are not using a real time Linux operating system, we can see some deviations occurring in figure 5.63. However they are quite small and don't have a considerable impact on the video quality. In figures 5.64, 5.65 and 5.66, 10 Mbps background traffic (VBR, 50ms-50ms and 100ms-100ms, buffer size 5 packets and 100 packets) is injected into the network causing heavy congestion. Under the heavy VBR traffic loading, the video frame inter-arrival time deviates considerably from 33ms, scattering between 0 ms and 140 ms. It is evident that frame inter-arrival times (buffer size of 5 packets) under 50ms-50ms VBR experiences smaller deviations as compared to the frame inter-arrival

time under 100ms-100ms VBR. Long buffer size (100 packets) seems to provide an advantage that lead to the reduction of the deviation of frame inter-arrival time experiences as seeing by comparing figures 5.66 and 5.65. This result is in agreement with the earlier PSNR analysis. The explanation for these phenomenons is that VBR traffic makes the frame jitter larger and irregular compared with CBR case shown in figure 5.31, and delays the video packet arriving at the receiver side on time. Therefore this will delay the reconstruction of the video frame that the delayed packet belongs to. As we know, the frame jitter is related to the video quality, so if the frame inter-arrival time is shown as in Fig 5.64, 5.65 and 5.66, the video quality is unacceptable to viewers.

Figures 5.67 to 5.70 present the detailed results of average frame inter-arrival time and its standard deviation with different buffer sizes (5,10, 100, 1000 packets) and different background VBR traffic loading (10ms-10ms, 50ms-50ms, 100ms-100ms). Because small buffers produce more packet losses, thus more frames are lost. The frame inter-arrival time increases between frames that have lost frames between them, increasing average values of frame inter-arrival time and standard deviation. For long burst VBR traffic (50ms-50ms, 100ms-100ms), the frame inter-arrival time largely deviates from the 33.3ms compared with short burst VBR traffic (10ms-10ms), as well as its standard deviation. However, the long buffer handles long burst VBR traffic better than short burst one because it let system have enough space and time to hold and process the packets. Another phenomena need to be noticed is that for small buffers, the short On-Off periods VBR traffic gives lower frame inter-arrival time, however, this reverses when the buffer becomes long. This behavior is same as the packet and frame losses behavior we discussed earlier.

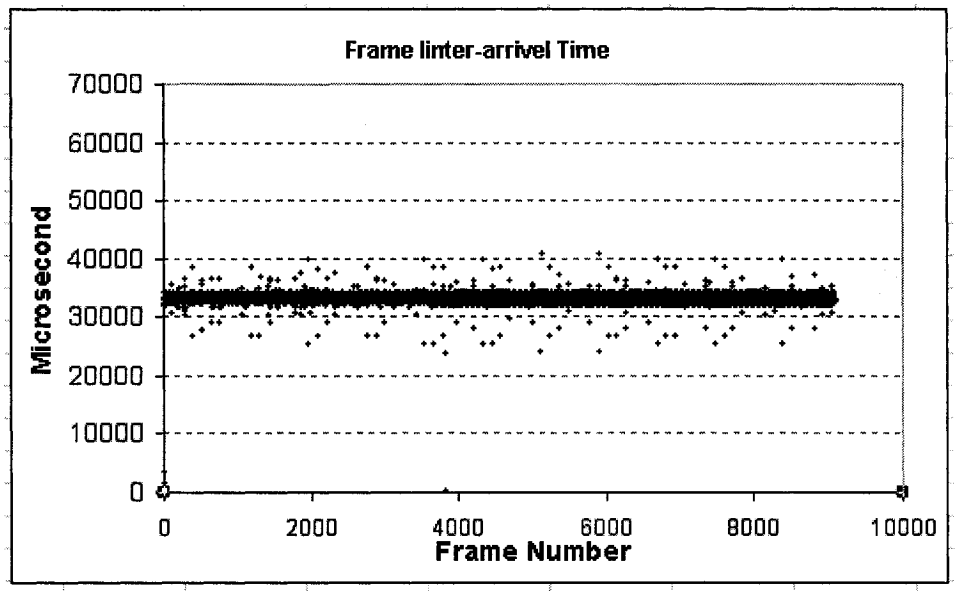


Figure 5.63 Frame Inter-arrival Time recorded during an experiment without background traffic.

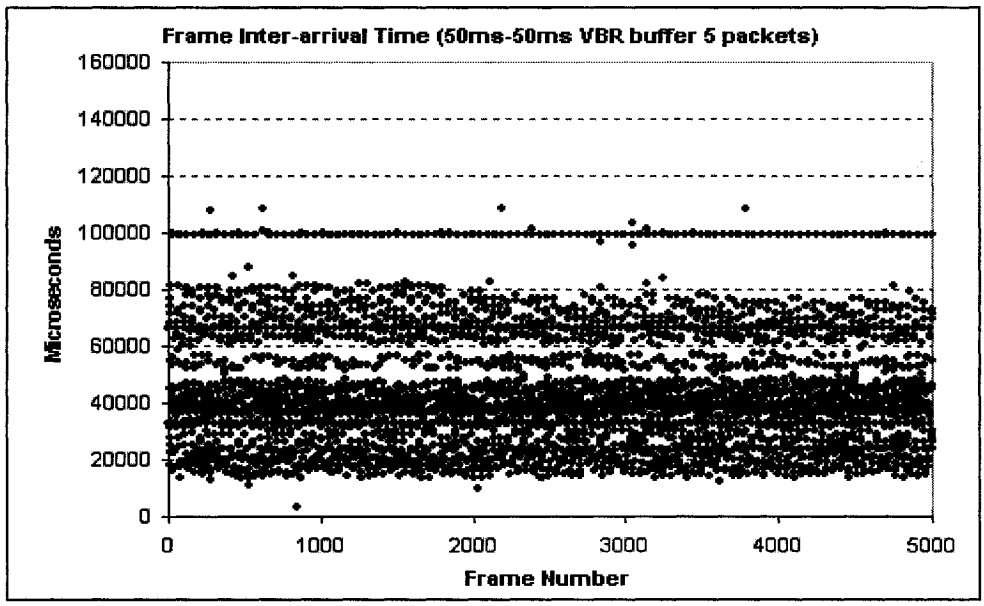


Figure 5.64 Frame Inter-arrival Times recorded during an experiment with VBR (50ms-50ms) background traffic loading 10 Mbps and buffer size at the core router of 5 packets

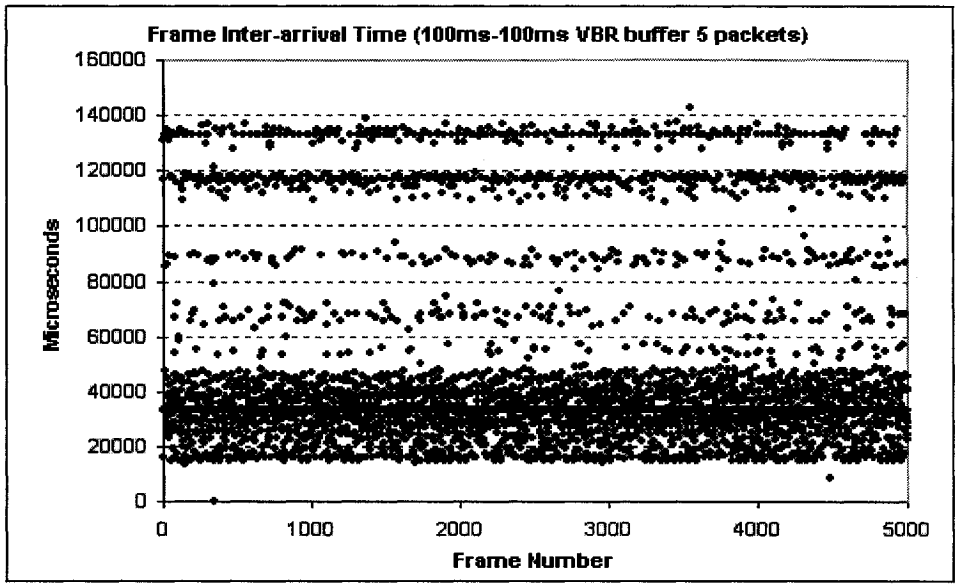


Figure 5.65 Frame Inter-arrival Times recorded during an experiment with VBR (100ms-100ms) background traffic loading 10 Mbps and buffer size at the core router of 5 packets

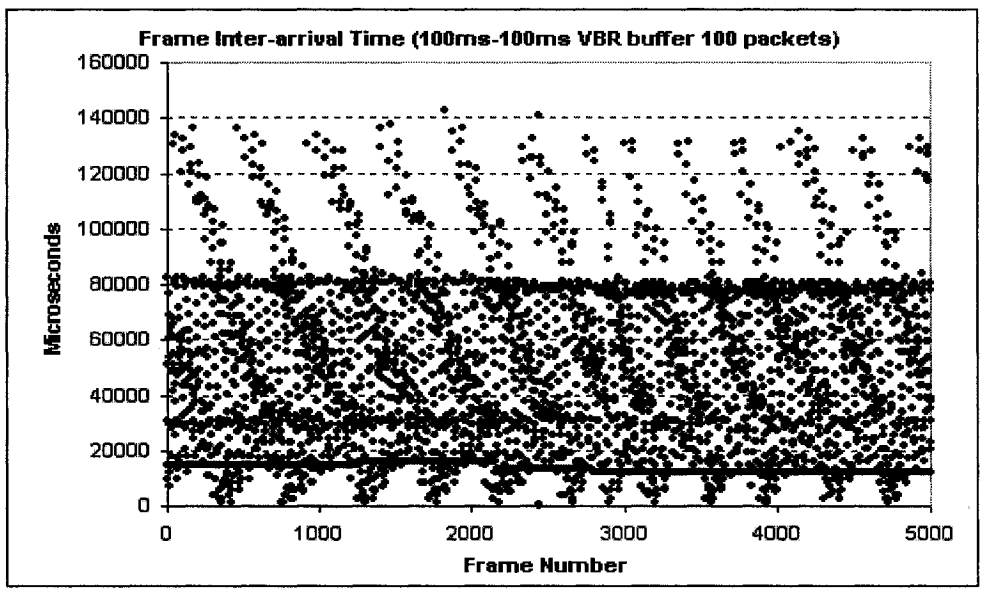
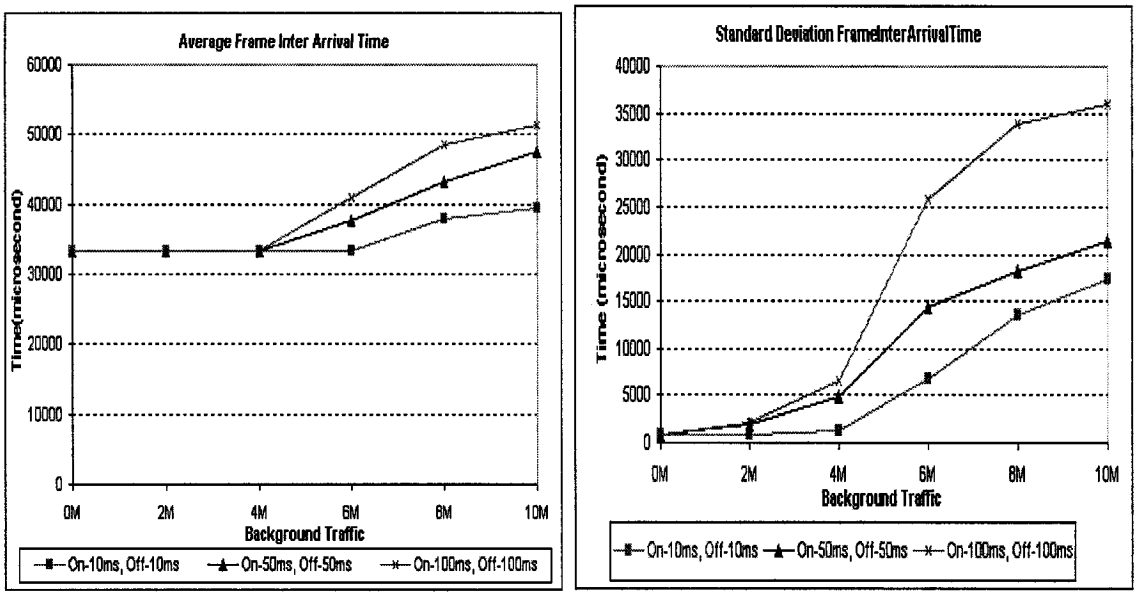
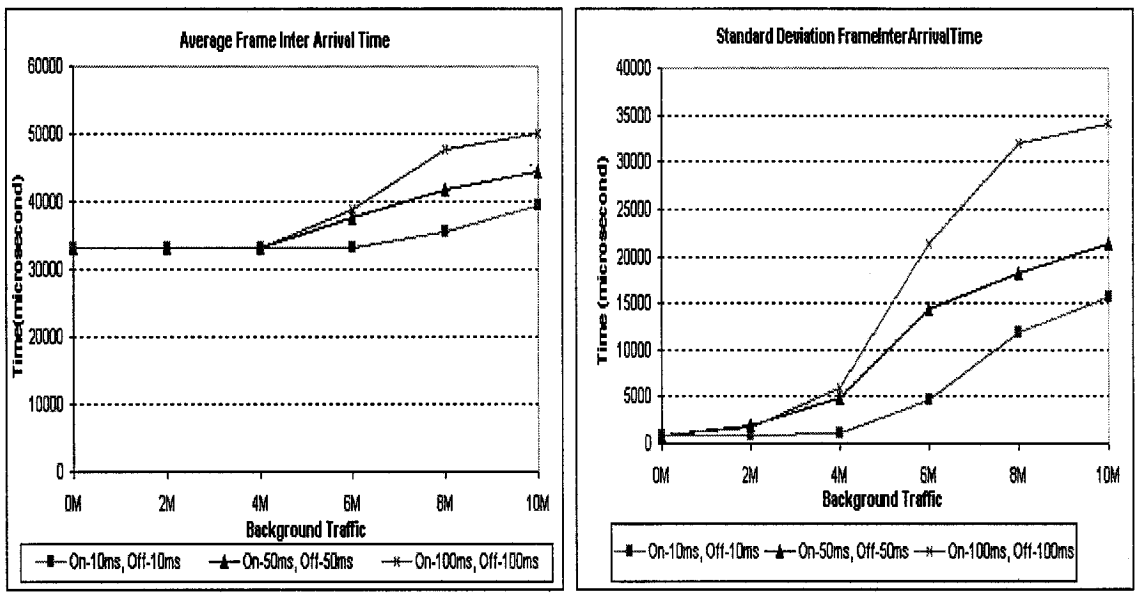


Figure 5.66 Frame Inter-arrival Times recorded during an experiment with VBR (100ms-100ms) background traffic loading 10 Mbps and buffer size at the core router of 100 packets



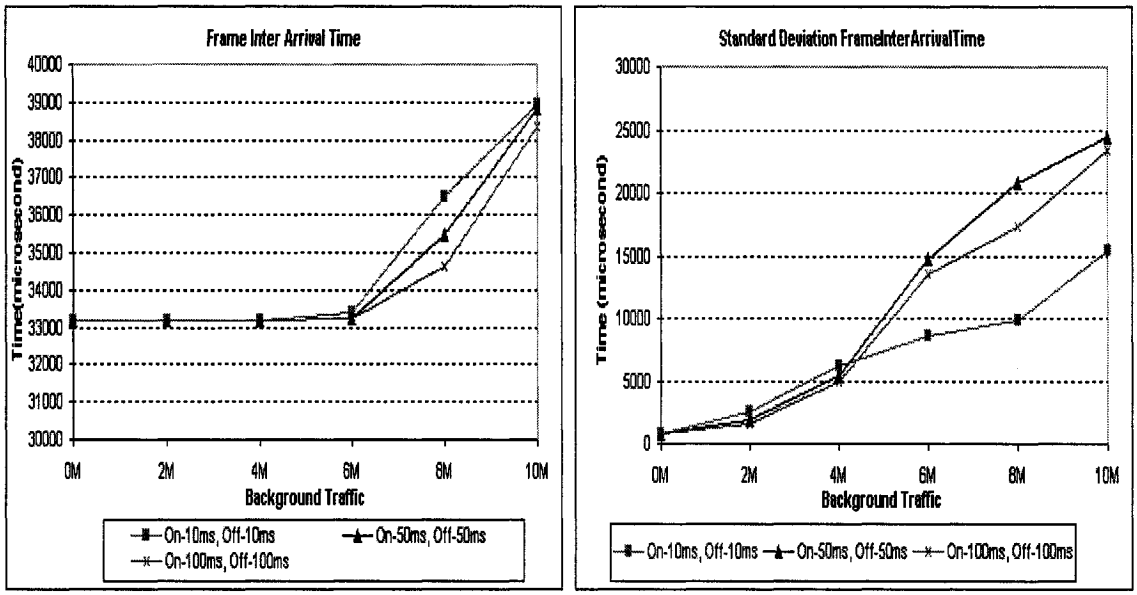
(a) (b)

Figure 5.67 (a) Average Frame Inter-Arrival Time and (b) standard deviation versus traffic load (5 packets buffer size)



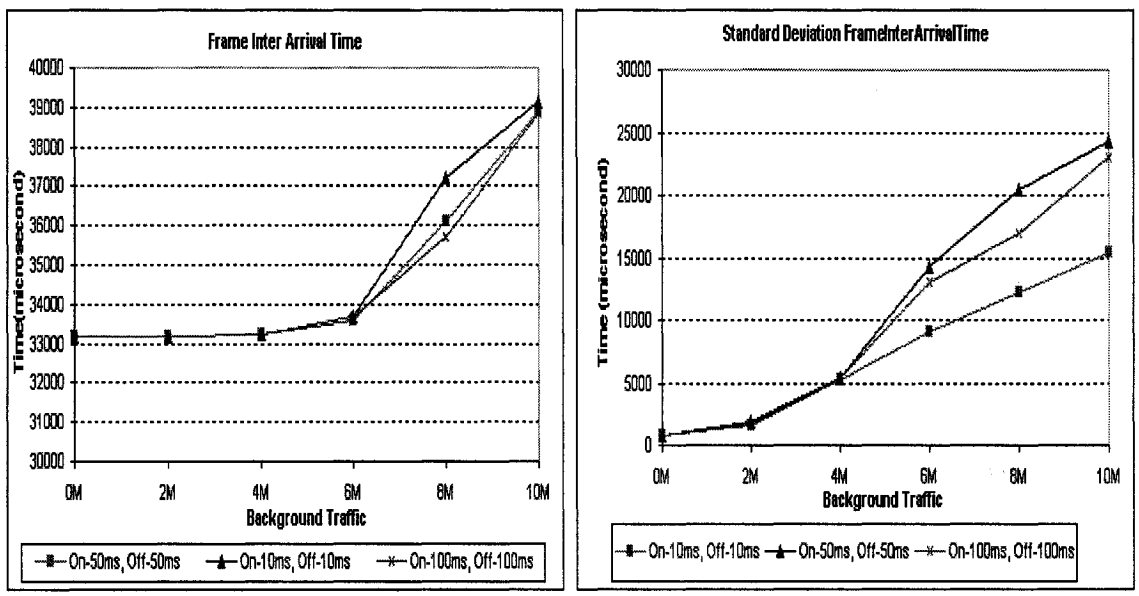
(a) (b)

Figure 5.68 (a) Average Frame Inter-Arrival Time and (b) standard deviation versus traffic load (10 packets buffer size)



(a) (b)

Figure 5.69 (a) Average Frame Inter-Arrival Time and (b) standard deviation versus traffic load (100 packets buffer size)



(a) (b)

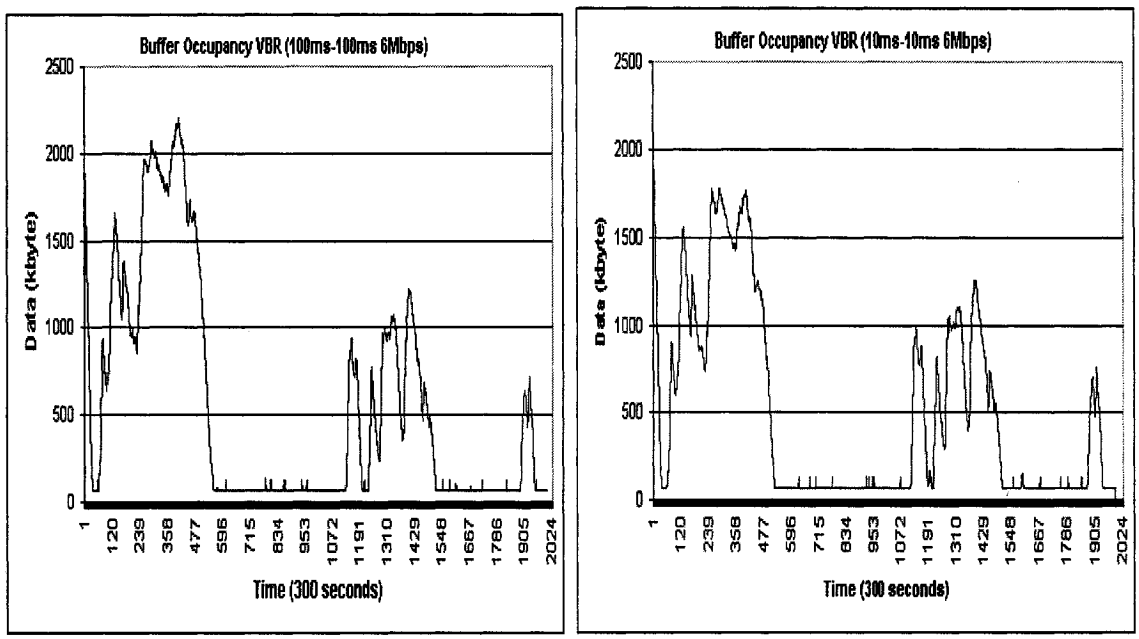
Figure 5.70 (a) Average Frame Inter-Arrival Time and (b) standard deviation versus traffic load (1000 packets buffer size)

As we discussed in the last section, sustainability of the connection is critical to our video application and directly linked to the buffer occupancy at the client side. In video

application, sustainability of the connection is reflected directly by the buffer occupancy at the client side. The decoder decodes the contents in the buffer. If there is no data in the buffer, the decoder experiences starvation and stops playing the video. It is evident that buffer occupancy is an important factor to the normal operation of a video application. Thus, the amount of video data stored in the client buffer has been monitored during the transmission of the video-clips. The measurements are done at the times when the decoder device requests 64KB of data to be extracted from the buffer.

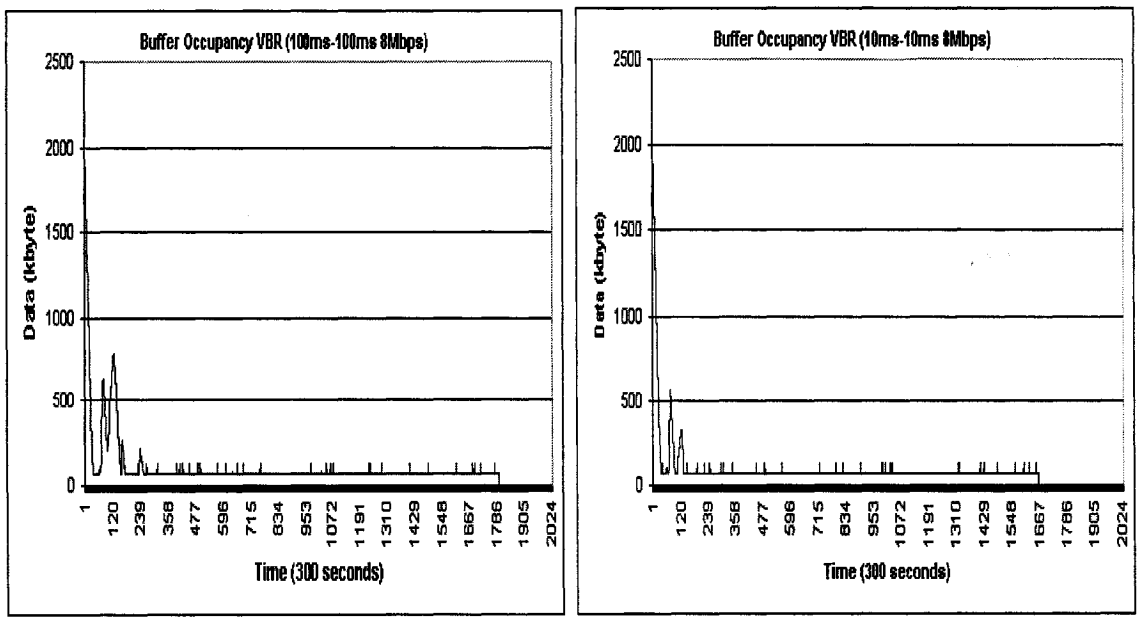
In order to understand the impact of VBR traffic loading, we present figures 5.71 to 5.73 recorded during several experiments. The buffer at the core router has buffer size of 100 packets. From earlier results related to packet loss rate and PSNR analysis, we know that for buffer size of 100 packets, serious packet loss starts around 6Mbps. We can see the big loss of buffer occupancy occurring. The peak of buffer occupancy recorded for 100ms-100ms VBR is only around 2,200 Kbytes and for 10ms-10ms only around 1,800K bytes, compared with the peak around 2,700K bytes shown in figure 5.33, which is sort of "benchmark" graph for buffer occupancy. At the same time, the graph is lower than that of figure 5.44 because some video packets are dropped at the core router and do not reach the client side. It is evident that 10ms-10ms VBR is worse than 100ms-100ms VBR for buffer size 100 packets and makes the decoder to reach the starvation point earlier. In figure 5.72a and 5.72b, the curves seem similar, however, the starving point for 10ms-10ms VBR occurs at 246 seconds, for 100ms-100ms at 264 seconds. These results are in agreement with the earlier PSNR, packet loss rate and frame inter-arrival time results and their analysis. The curves in figure 5.74 and 5.75 present the average buffer occupancy under different buffer size and VBR traffic loading. In the short buffer size of 5 packets, the buffer occupancy begins to decrease, dropping from around 600 Kbytes to around 270 Kbytes and 230 Kbytes for 100ms-100ms and 50ms-50ms VBR at traffic loading 4Mbps, and around 250K bytes for 10ms-10ms VBR at traffic loading 6Mbps. Comparing the figures, we find that long buffer size has better performance than short buffer in terms of buffer occupancy, because most video packets are dropped at the core router when buffer size is short, thus the decoder at the client side cannot get enough data to play and quickly reach the starvation. In addition, we find the average buffer occupancy of 10ms-10ms

VBR is bigger than that of 50ms-50ms and 100ms-100ms with small buffer size and smaller with large buffer size. Because buffer occupancy is related to the packet dropping behaviour at the core router, from the earlier packet loss analysis, we can conclude that the results of buffer occupancy are in agreement with the earlier packet loss rate results.



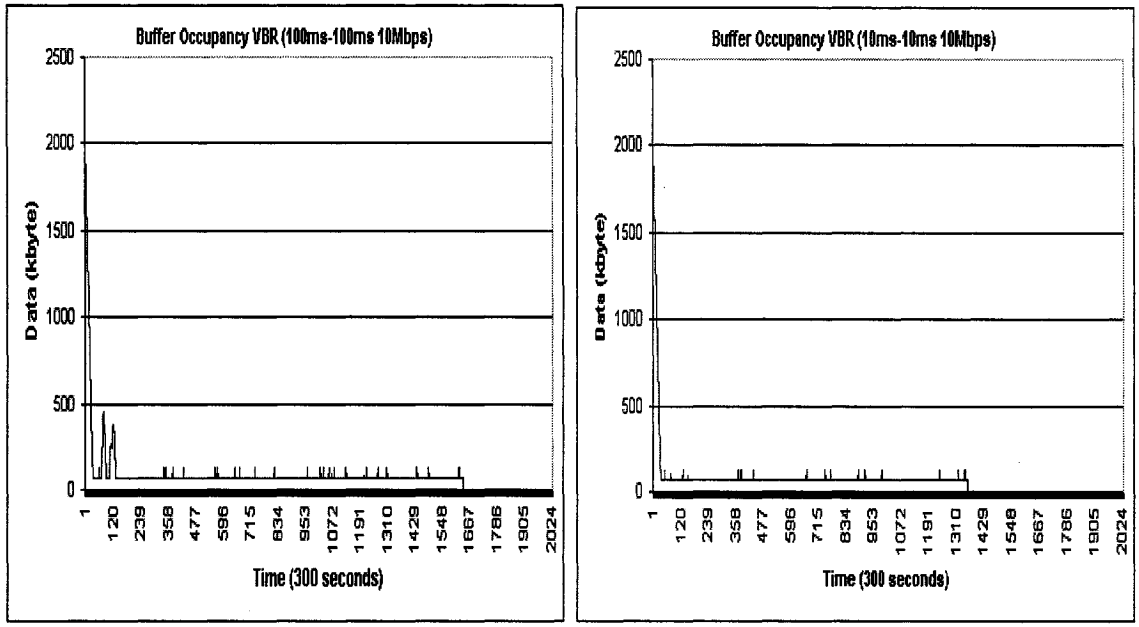
(a) (b)

Figure 5.71 Buffer Occupancy at the decoder with VBR background traffic of 6Mbps: (a) 100ms-100ms and (b) 10ms-10ms



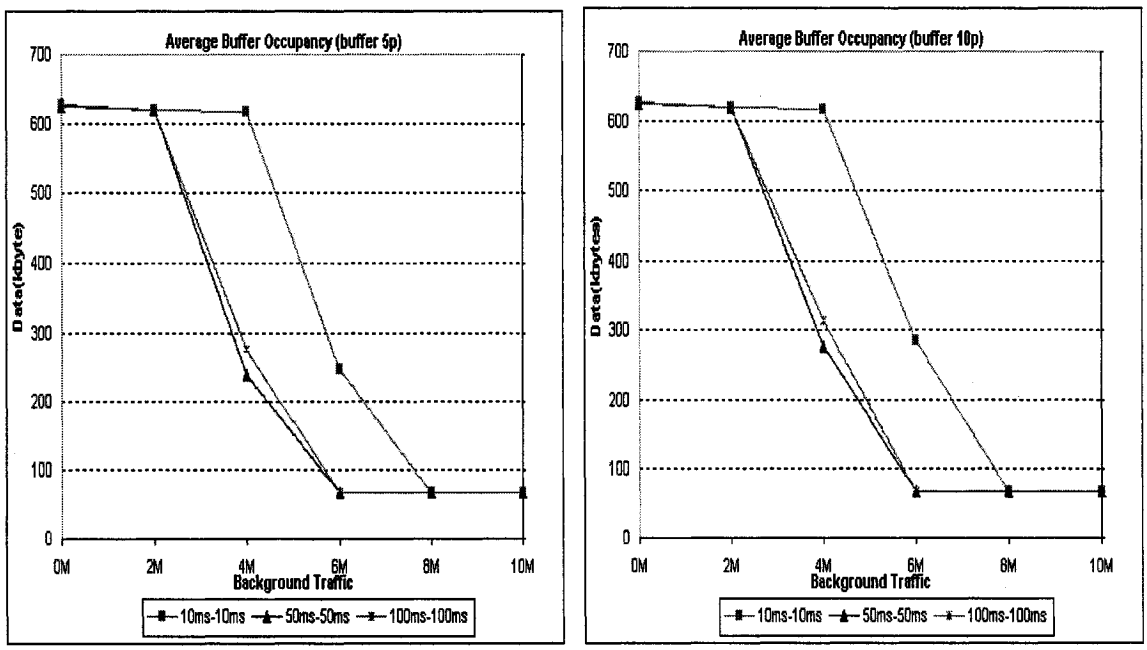
(a) (b)

Figure 5.72 Buffer Occupancy at the decoder with VBR background traffic of 8Mbps: (a) 100ms-100ms and (b) 10ms-10ms



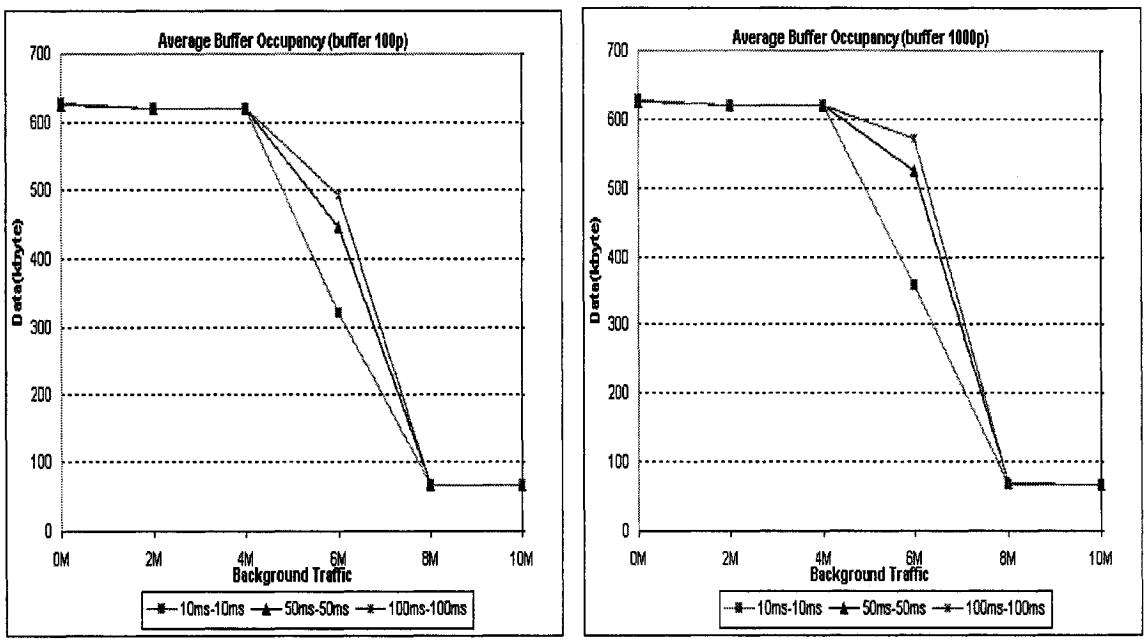
(a) (b)

Figure 5.73 Buffer Occupancy at the decoder with VBR background traffic of 10Mbps: (a) 100ms-100ms and (b) 10ms-10ms



(a) (b)

Figure 5.74 Average Buffer Occupancy versus traffic loading (a) buffer 5 packets and (b) buffer 10 packets



(a) (b)

Figure 5.75 Average Buffer Occupancy versus traffic loading (a) buffer size of 100 packets and (b) buffer size of 1000 packets

It is evident that, even different buffer size used, the various VBR background traffic loadings still severely affect the performance of video application.

5.2.4 Video performance under Differentiated Services

As mentioned earlier, the traditional Best Effort service cannot guarantee the quality of the video application, especially in highly congested network environment. Therefore, the DiffServ QoS architecture is introduced by the IETF to resolve this weakness. However, Differentiated Services provide soft "guarantees" to the service. Thus, it is important to understand how these guarantees affect the performance of the applications. In addition, we need to understand the impact of the different traffic forwarding, scheduling and queuing disciplines. As described in Chapter 4, the Priority Queuing (PQ) and the Class Based Queuing (CBQ) are used for scheduling the traffic inside DiffServ.

In the following sections, the performance of MPEG2 video in a DiffServ network with CBR and VBR traffic is analyzed.

We present results for two different packet-forwarding policies used at the DiffServ core router. These are PQ and CBQ. A 4 MB buffer was used at the client. Video traffic is classified as Expedited Forwarding (EF). In the PQ configuration, EF has the highest priority. In the CBQ configuration, EF was allocated 4 Mbps, while best effort traffic used the remaining 6 Mbps. Background traffic consists of *udp* packets.

The CBQ is a hierarchical link-sharing algorithm for managing the network bandwidth resources by performing the resource reservation. With CBQ, a specific class can be allocated certain bandwidth. If a class does not produce traffic volumes at the level of its resource allocation, it can allow other classes (producing traffic at levels higher than their allocation) to consume the unused portion of the resource. CBQ is a powerful traffic-scheduling algorithm, however, it is computing processing intensive.

The buffer size is an important factor for the performance of video application. As discussed in the last two sections, small buffer size will not allow the router to absorb a sudden surge of traffic, large buffer space will increase the queuing delay within the core

router, in order to address these issues, we contacted tests with different buffer sizes and analyzed the impact.

Experiments are contacted with the GN Nettest traffic generator producing Ethernet frames of 1500 bytes size. All experiments are conducted for the following background traffic loads:

- 0 Mbps
- 2 Mbps
- 4 Mbps
- 6 Mbps
- 8 Mbps
- 10 Mbps

We are conducting the experiments for four different values of EF buffer size at the core router:

- 2 packets (7.5 Kbytes)
- 5 packets (15 Kbytes)
- 15 packets (150 Kbytes)
- 1000 packets (1500 Kbytes)

The following parameters are measured:

- Packet loss rate at core router
- Forwarding delay at the core router
- Number of video frames lost
- PSNR value of the individual frames
- Frame inter-arrival time.

Experiments are performed 8 times for each scenario. The collected statistics are processed, and the following statistical parameters are calculated:

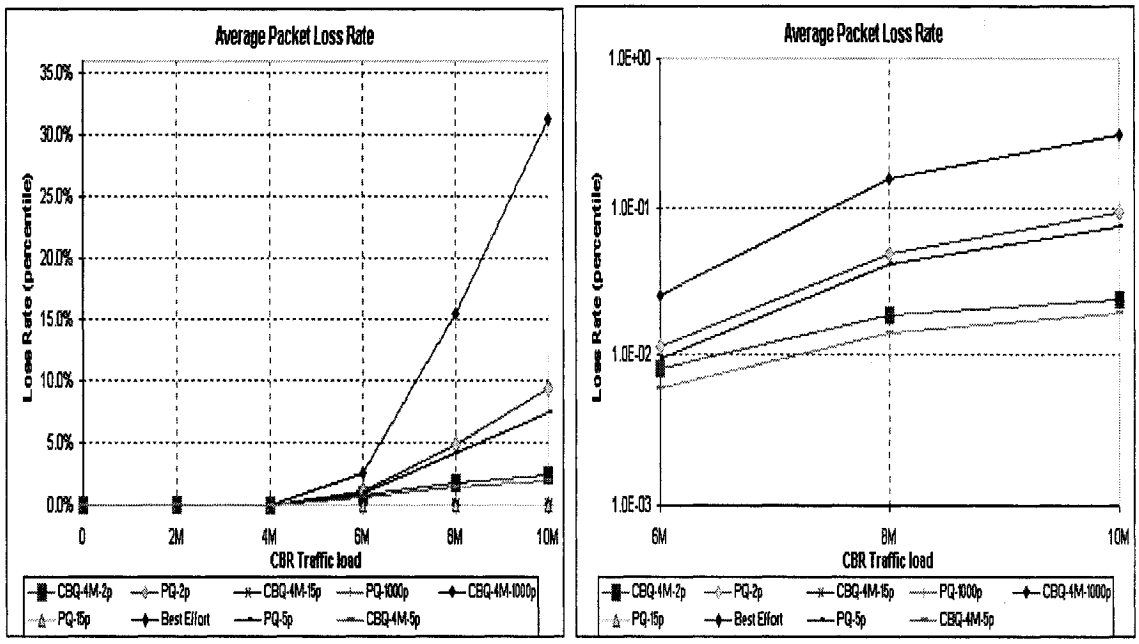
- Average packet loss rate at the core router
- Average forwarding delay at the core router
- Standard deviation of average forwarding delay at the core router

- Average PSNR value
- Standard deviation of PSNR value
- Average frame inter-arrival time.
- Standard deviation of frame inter-arrival time

5.2.4.1 Differentiated Services under CBR background Traffic

As mentioned earlier, the CBQ and PQ are used in DifffServ for traffic forwarding and queuing policies to provide a desirable QoS guarantee for the video application. With the different buffer size we defined earlier, DiffServ mechanism presents corresponding results when background traffic loading is applied. The packet loss rate experienced by the various configurations is shown in figure 5.76. For both CBQ and PQ, no video packet losses were encountered, when the EF buffer size is 15 packets or larger. The improvement offered by DiffServ over best effort becomes higher, as the network load increases. If no DiffServ, the packet loss rate will jump to 32% and the quality of video application will be intolerable. However, we still see packet loss happening when the EF buffer size is small. When the EF buffer size is 2 packets, the PQ's packet loss rate is around 9%, the CBQ's packet loss rate is around 2.5%. When the EF buffer size is 5 packets, the PQ's packet loss rate is decreased to 7.5%, the CBQ's packet loss rate is around 2%.

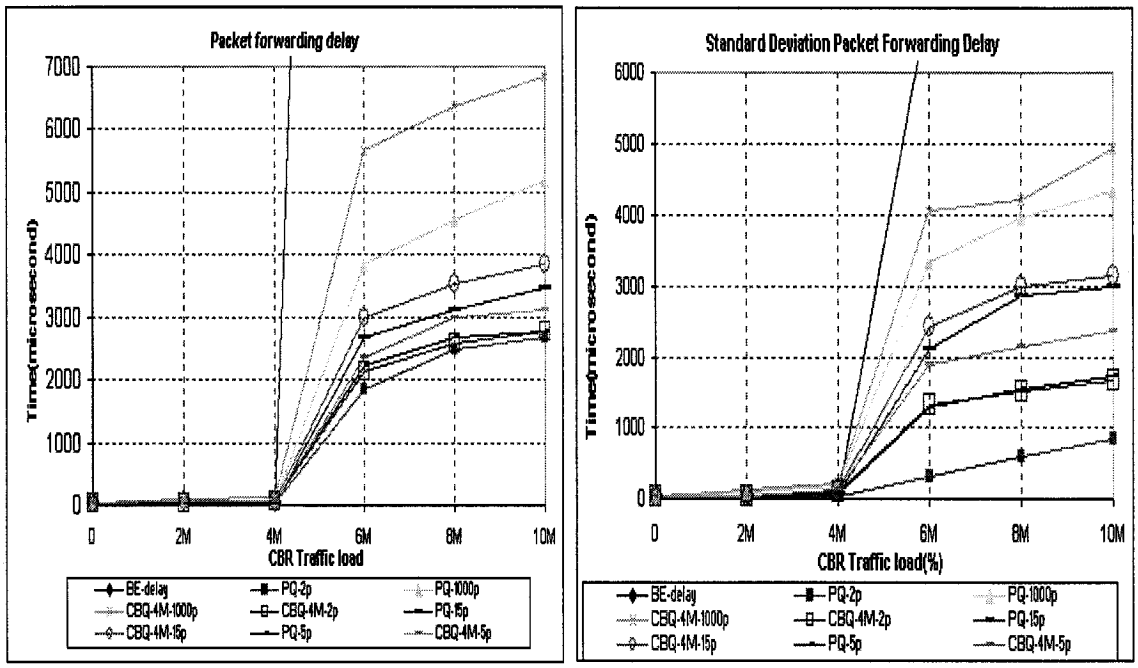
For small buffer size, CBQ and PQ have different performance. When the background traffic becomes high, the best effort (BE) queue is not empty, there is always at least a packet to be served inside the buffer. Therefore, if the EF queue is emptied, the server becomes busy for processing the BE packets. If more video packets arrive at this moment, they will be lost, due to the small size of the queue. This explains why PQ has a relatively high increase of packet losses at the higher network loading. However, if we make the BE packet size smaller, the performance will be improved and packet loss rate will decrease [HYU01]. CBQ is a resource sharing algorithm, it isolates EF traffic from BE traffic, therefore, the video stream is less affected by the increase in the BE traffic, as well as the packet size.



(a) (b)

Figure 5.76 Average Packet loss rate (a) Linear (b) Logarithmic graph

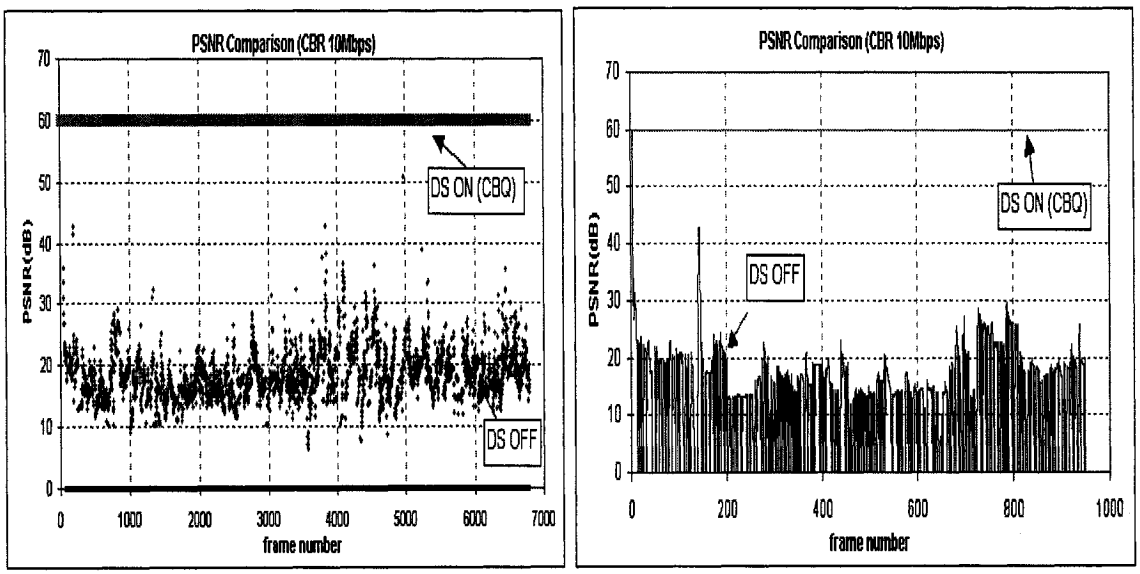
Figure 5.77a presents the average packet forwarding delay that video packets experienced in the core router. It is evident that short buffer size produces small queuing delay. Curves show that PQ provides lower delay compared with CBQ because PQ has higher packet losses, this means that more packets are lost under congestion and they don't count in the average delay. CBQ keeps more packets under congestion and since theses packets have higher delay, the average delay increases. Similar rule can also applied to buffer case, small buffer versus larger buffer, smaller buffer has more packets losses and less average delay. Figure 5.77b presents the standard deviation of the packet forwarding delay. We can see that the trend is similar to that of average packet forwarding delay.



(a) (b)

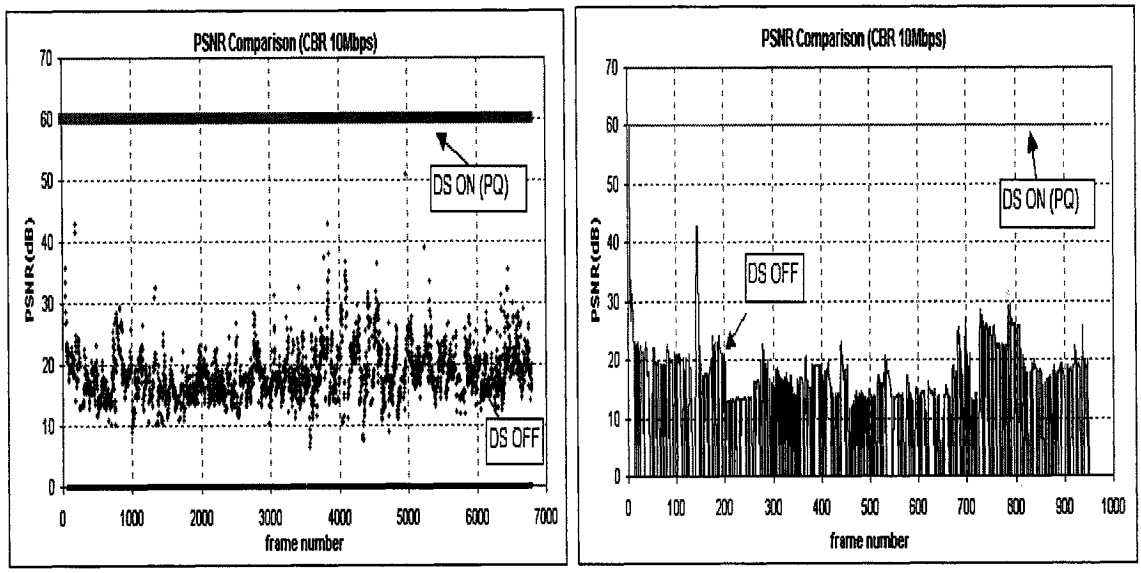
Figure 5.77 (a) Average Packet Forwarding Delay and (b) its standard deviation

As a convincing index for evaluating the image quality, no doubt, the PSNR will provide reliable proof in terms of assessing the effectiveness of the DiffServ mechanism. First let us see figure 5.78, 5.79, the curves present the PSNR value with EF buffer size 15 packets while DiffServ is enabled and with buffer size 100 packets while DiffServ is disabled (Best Effort) under CBR 10Mbps. It is evident that CBQ, PQ schemes all get desirable performance, PSNR = 60dB, with buffer size 15packets. In best effort case, though the buffer size is 100 packets, most of PSNR values are below 30dB as shown in figure 5.78, 5.79. This result is identical to what we analyzed above, 0% packet loss rate means that the original video is reconstructed at the client without any content lose, PSNR value is 60dB.



(a) (b)

Figure 5.78 (a) PSNR comparison graph between CBQ, EF buffer size 15 packets and Best Effort under CBR 10Mbps (b) Detailed first 1000 frames PSNR

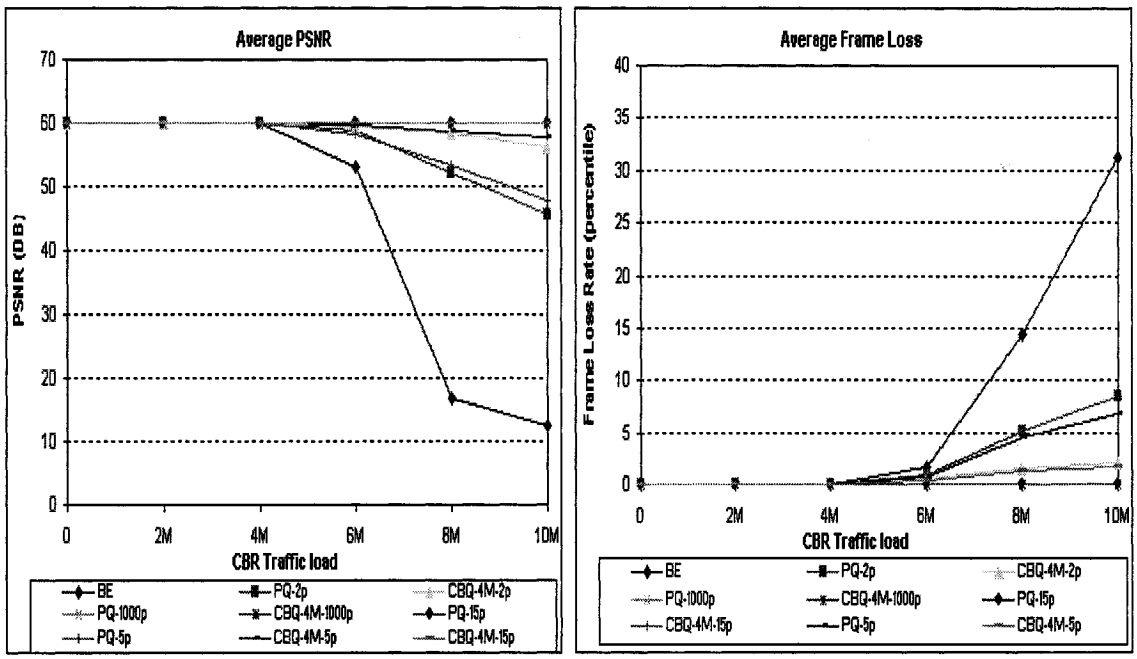


(a) (b)

Figure 5.79 (a) PSNR comparison graph between PQ, EF buffer size 15 packets and Best Effort under CBR 10Mbps (b) Detailed first 1000 frames PSNR

The curves in figure 5.80a present the PSNR value for all test scenarios. It is evident that, DiffServ mechanism, even with short buffer size 2 packets, has better performance than

best effort with buffer size 100 packets. In addition, CBQ also shows its strong ability of protecting video application when compared with PQ. The PSNR value is 56.3321dB for CBQ and only 45.6323dB for PQ when buffer size is 2 packets under CBR 10Mbps. However, CBQ and PQ with buffer size 15 packets or more all keep PSNR at 60dB. The frame loss analysis is also identical to our prior analysis as shown in figure 5.80b. There is no frame loss when suitable buffer size is chosen for EF class, even in short buffer case, the frame loss rate under DiffServ still significantly lower than that of under best effort with long buffer space.



(a) (b)

Figure 5.80 (a) Average PSNR and (b) Average Frame Loss graph

Figure 5.81a shows us the frame inter-arrival time under CBQ with buffer size 15 packets when traffic loading is CBR 10Mbps. By comparing figure 5.32 and 5.81a, it is evident that use of DiffServ has reduced the deviation of the frame inter-arrival times from the 33ms and values of frame inter-arrival time have been reduced considerably. It is almost identical to the case where no background loading exists. Figure 5.81b presents the frame inter-arrival time under CBQ scheme with buffer size 5 packets when traffic loading is

CBR 10Mbps. We can see that the performance retains very good. We also see the PQ performance in figure 5.82, compared with CBQ, its scattering scope is wider.

In order to develop an understanding how the network behavior affects the application, we present, in figures 5.83, 5.84, the frame inter-arrival time and its standard deviation data for all scenarios. It is evident that CBQ demonstrates robust behavior.

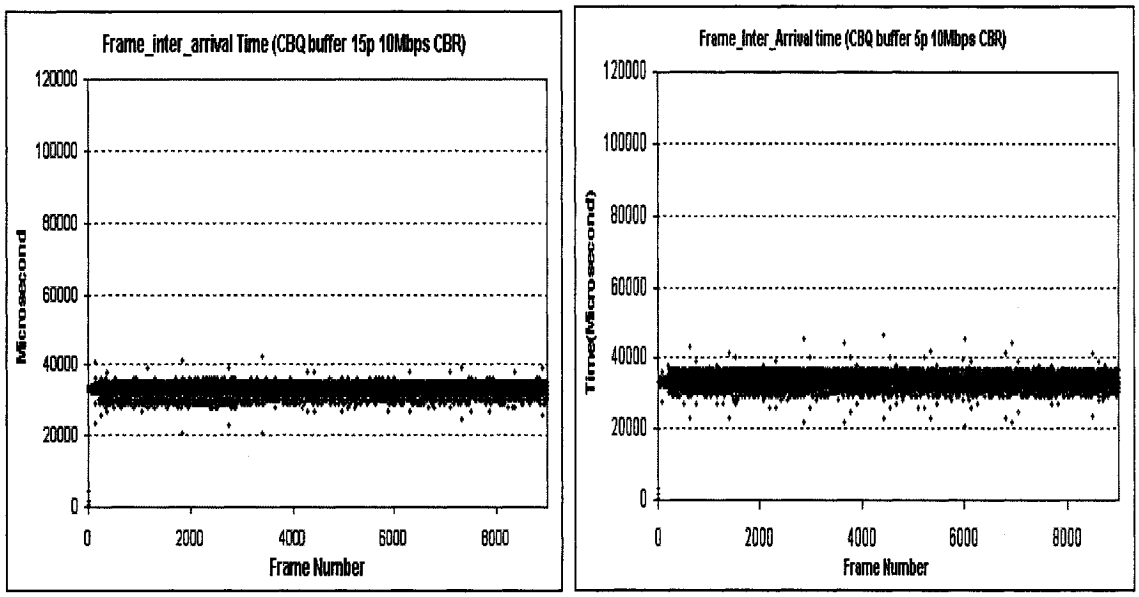
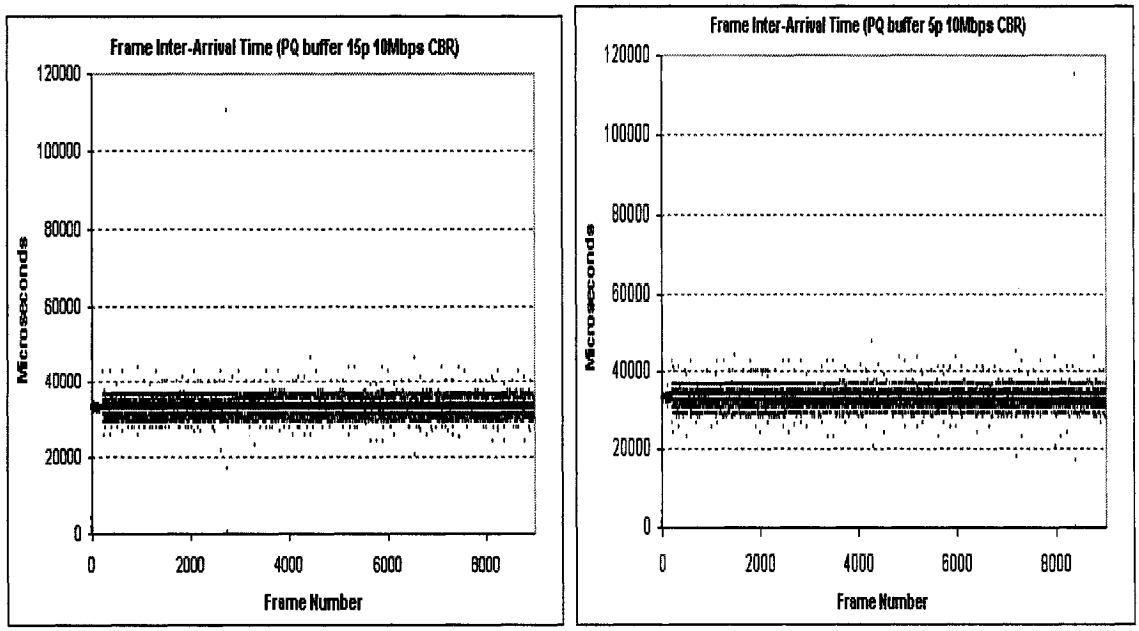
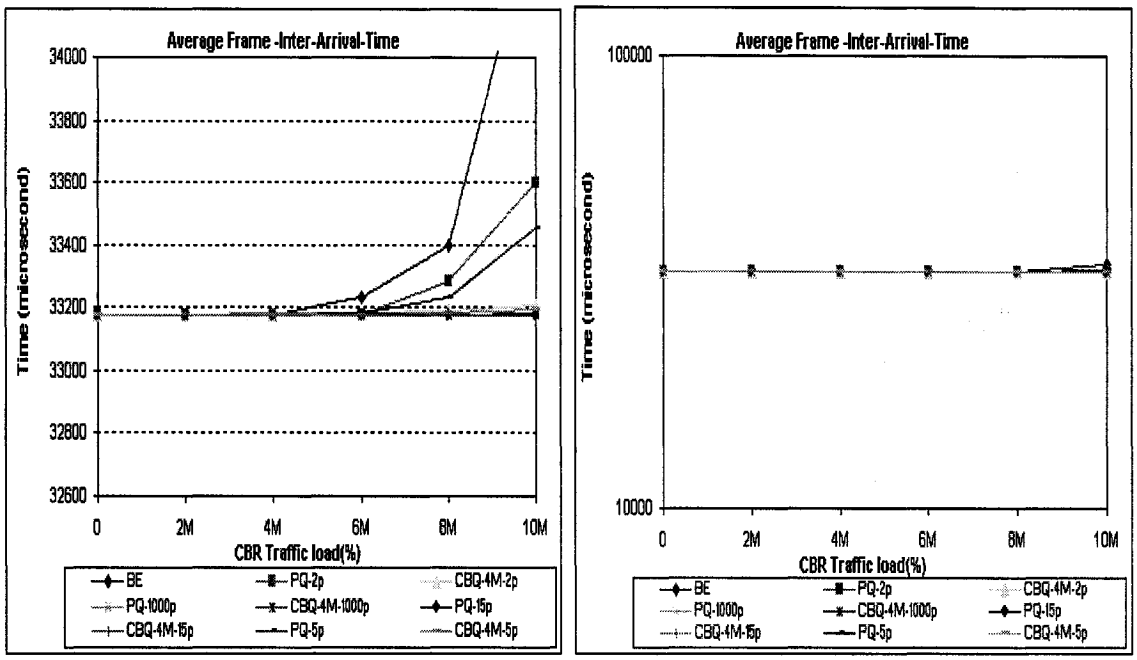


Figure 5.81 Frame inter-arrival time for CBQ when buffer size (a) 15 packets and (b) 5 packets under 10Mbps CBR background traffic.



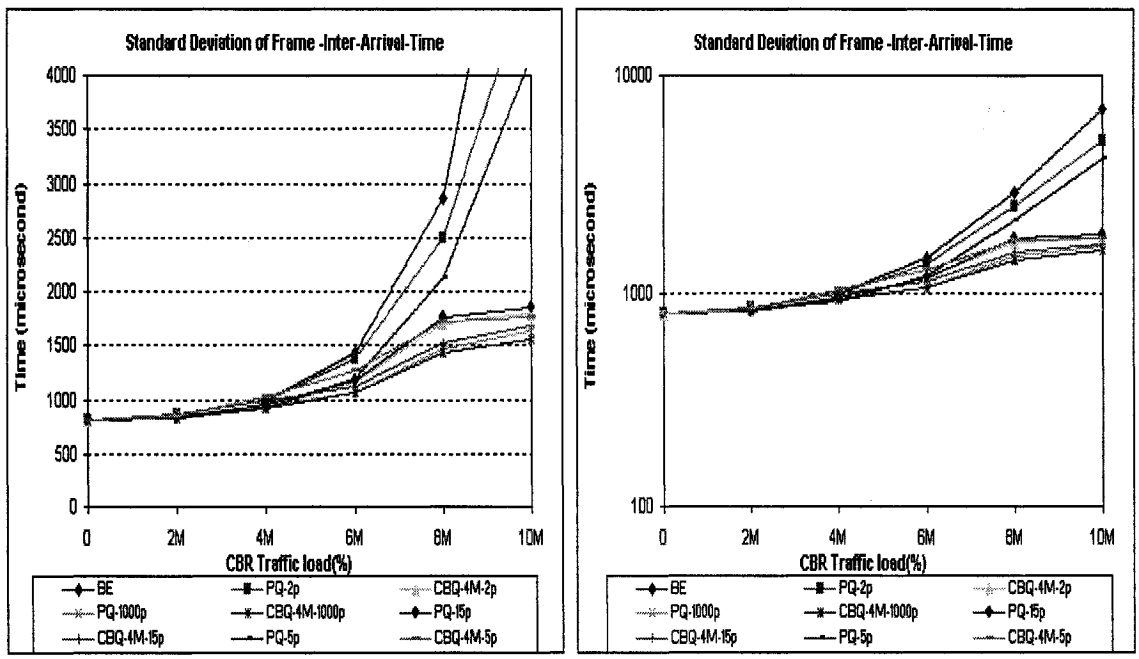
(a) (b)

Figure 5.82 Frame inter-arrival time for PQ when buffer size (a) 15 packets and (b) 5 packets under 10Mbps CBR background traffic.



(a) (b)

Figure 5.83 (a) Average frame inter-arrival time and (b) its logarithmic graph

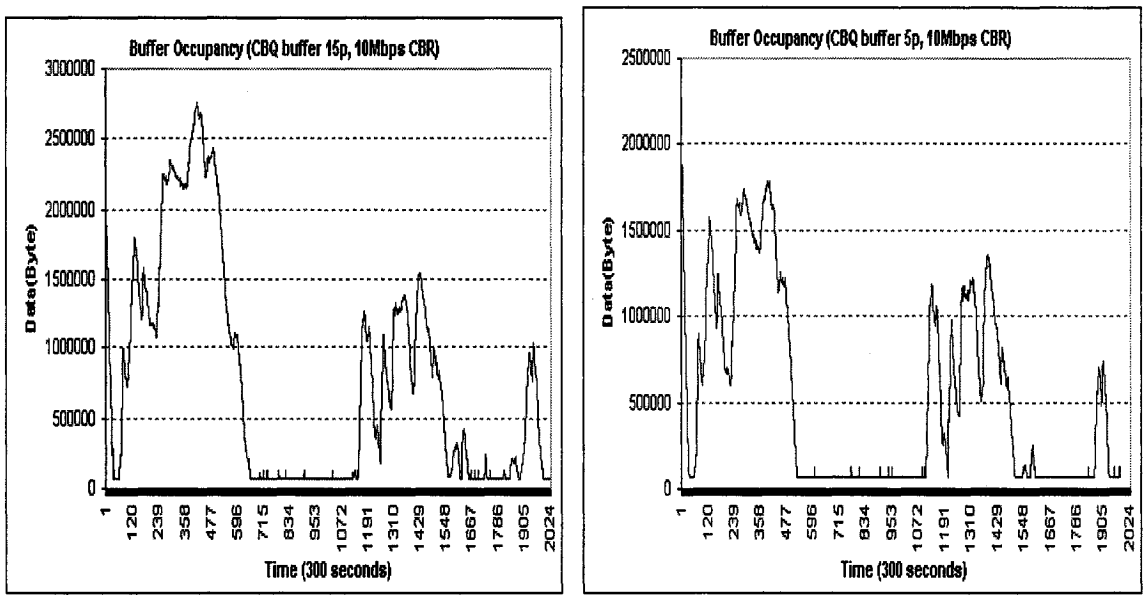


(a) (b)

Figure 5.84 (a) Standard deviation of frame inter-arrival time and (b) its logarithmic graph

It does not only keep the frame inter-arrival time around 33.2ms but also maintains the standard deviation below 2ms over the entire traffic loading scale. However, for best effort and PQ, frame inter-arrival time and its standard deviation increase significantly for higher traffic loading.

Sustainability of connection refers to the ability of the network to maintain the video connection until its end. The ability is protected by DiffServ even under high background traffic loading which severely damage the sustainability of the connection under best effort. The graph shape in figure 5.85a is identical to that in figure 5.34 when 0% background traffic loading is applied. This means that all video content is received by the client. Even for the short buffer size case, the performance has been highly improved as shown in figure 5.85b, when compared with 5.36. Figure 5.86 displays the average buffer occupancy under DiffServ.



(a) (b)

Figure 5.85 Buffer occupancy, (a) CBQ buffer size 15 packets, 10Mbps CBR and (b) CBQ buffer 5 packets, 10Mbps CBR.

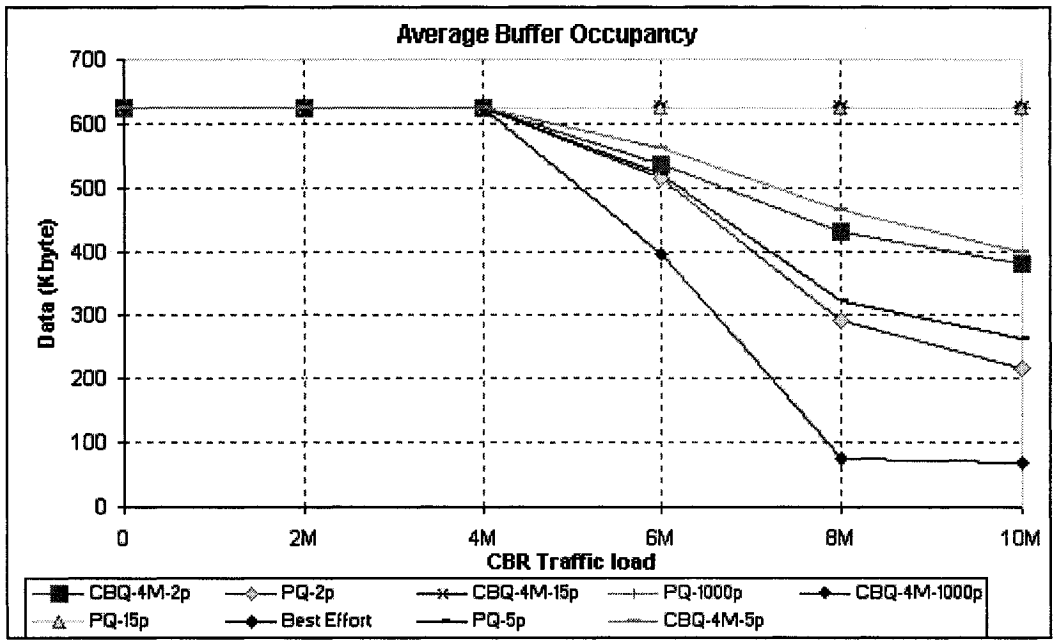


Figure 5.86 Average Buffer Occupancy (CBR)

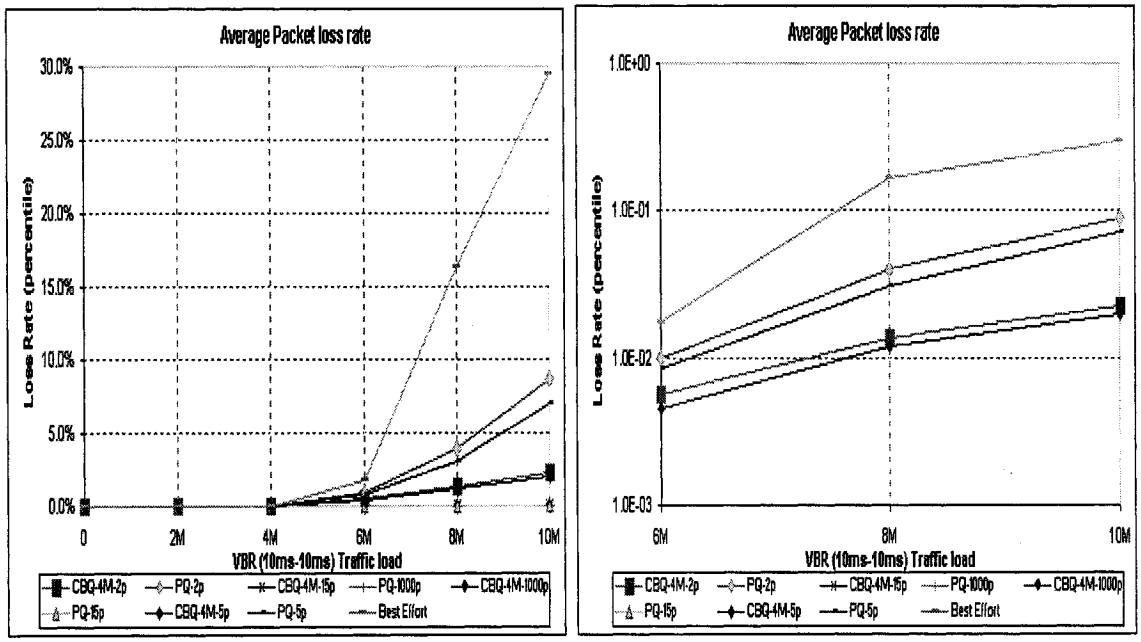
5.2.4.2 Differentiated Services under VBR background Traffic

In this set of experiments, the GN Nettest traffic generator is producing background traffic with an ON (active) OFF (inactive) pattern. The burst traffic parameters are as follows:

- ON 10ms, OFF 10ms
- ON 50ms, OFF 50ms
- ON 100ms, OFF 100ms

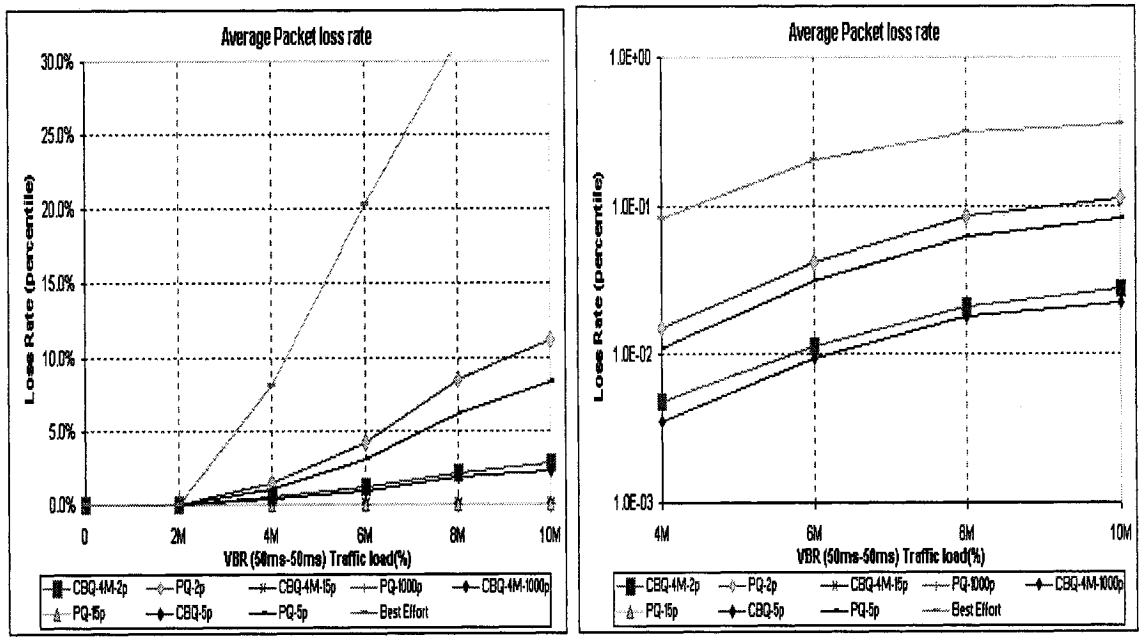
The experiments were run 8 times, sending 100,000 packets every time. This gives a total of 800,000 packets for every scenario.

Figures 5.87 to 5.89, present the testing results under DiffServ with VBR traffic loading (10ms-10ms, 50ms-50ms, 100ms-100ms) in terms of packet loss rate. It is evident that DiffServ with CBQ and PQ schemes and adequate EF buffer size (15 packets or more) provides high quality to video applications (very low packet and frame loss). In addition, DiffServ considerably improves the quality of video even for very short buffer size when compared to the best effort scenario.



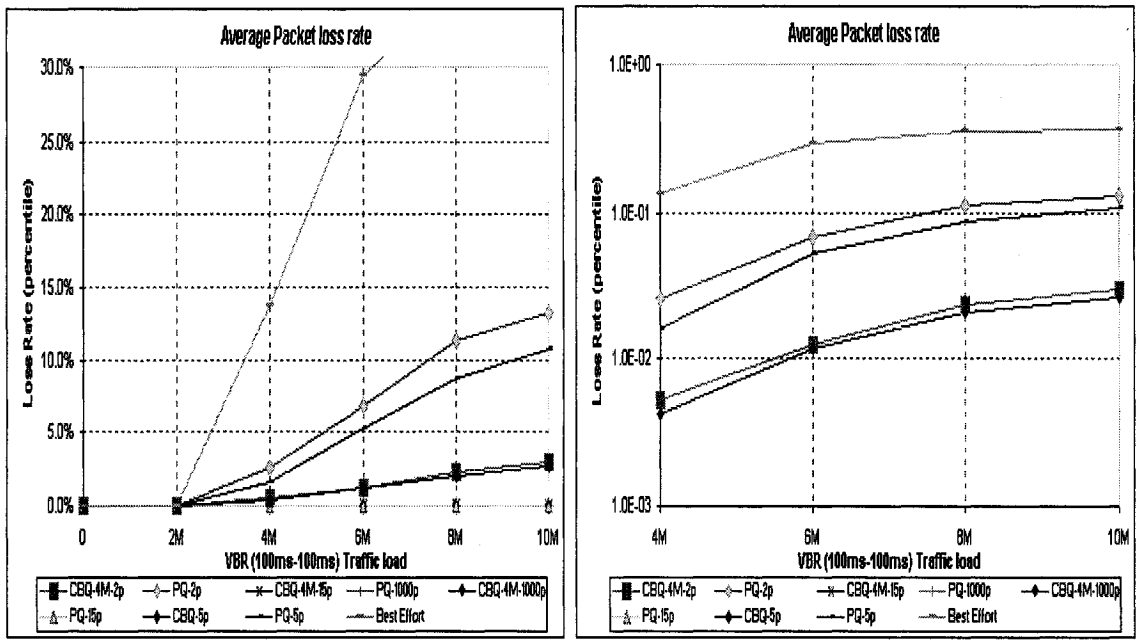
(a) (b)

Figure 5.87 (a) Average Packet loss rate and (b) its logarithmic graph (10ms-10ms VBR)



(a) (b)

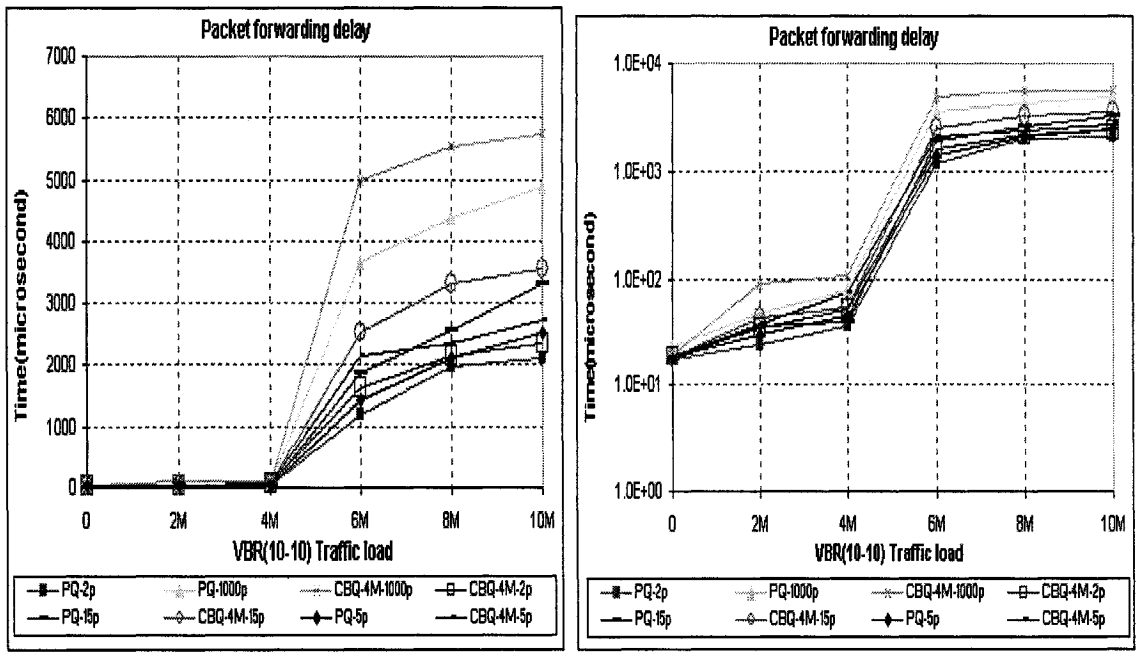
Figure 5.88 (a) Average Packet loss rate and (b) its logarithmic graph (50ms-50ms VBR)



(a) (b)

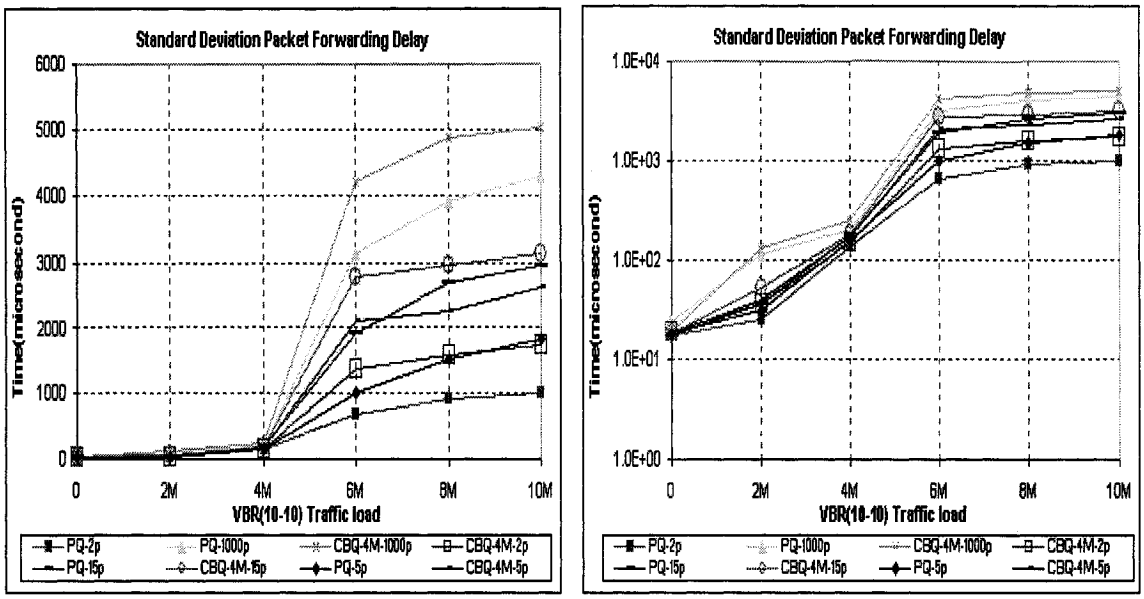
Figure 5.89 (a) Average Packet loss rate and (b) its logarithmic graph (100ms-100ms VBR)

Figures 5.90 to 5.95 present the packet forwarding delay and the standard deviation of packet forwarding delay results under DiffServ with VBR traffic loading (10ms-10ms, 50ms-50ms, 100ms-100ms). DiffServ considerably reduces the delay as well as standard deviation, thus improve the performance of video applications.



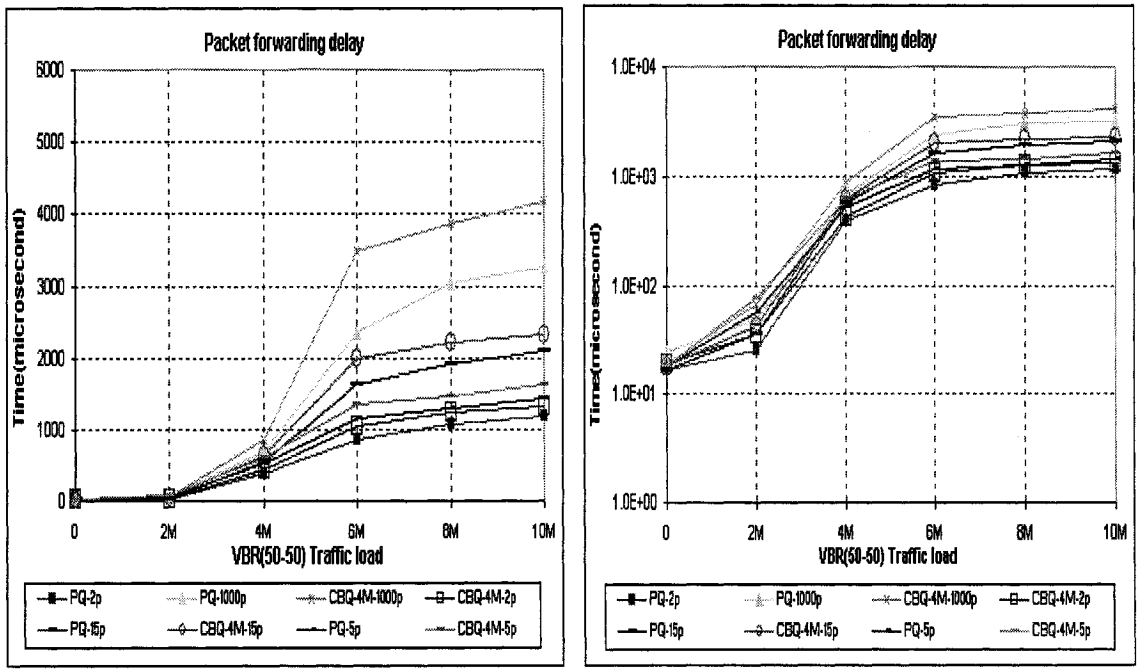
(a) (b)

Figure 5.90 (a) Average Packet Forwarding Delay and (b) its logarithmic graph (10ms-10ms VBR)



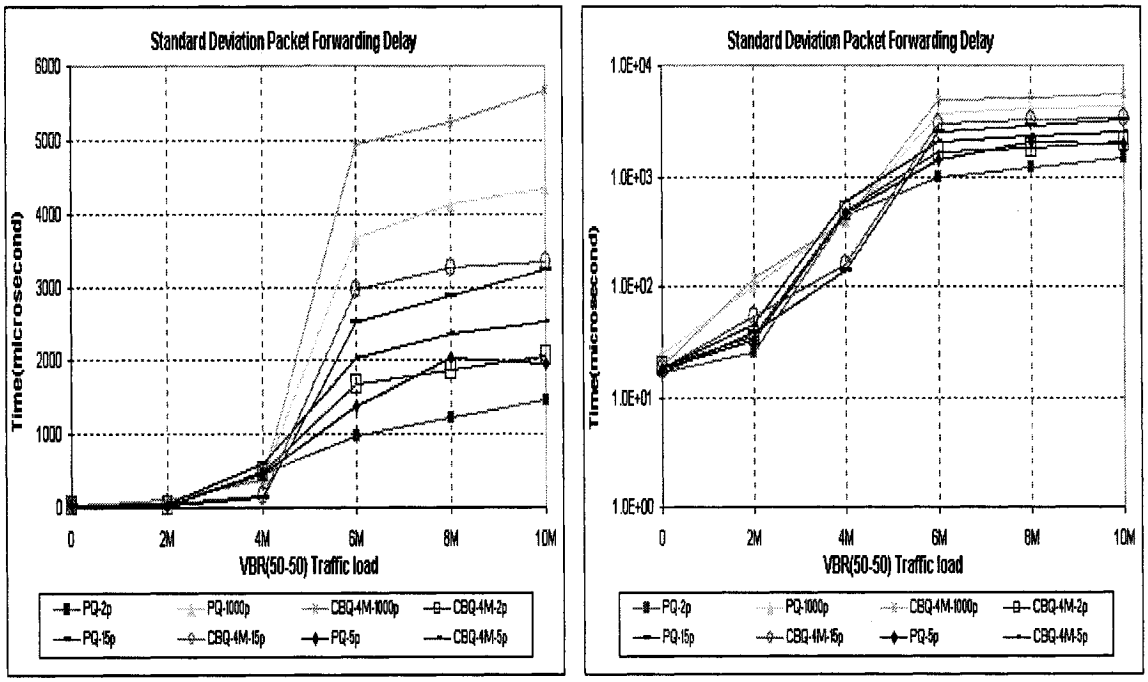
(a) (b)

Figure 5.91 (a) standard deviation of Packet Forwarding Delay and (b) its logarithmic graph (10ms-10ms VBR)



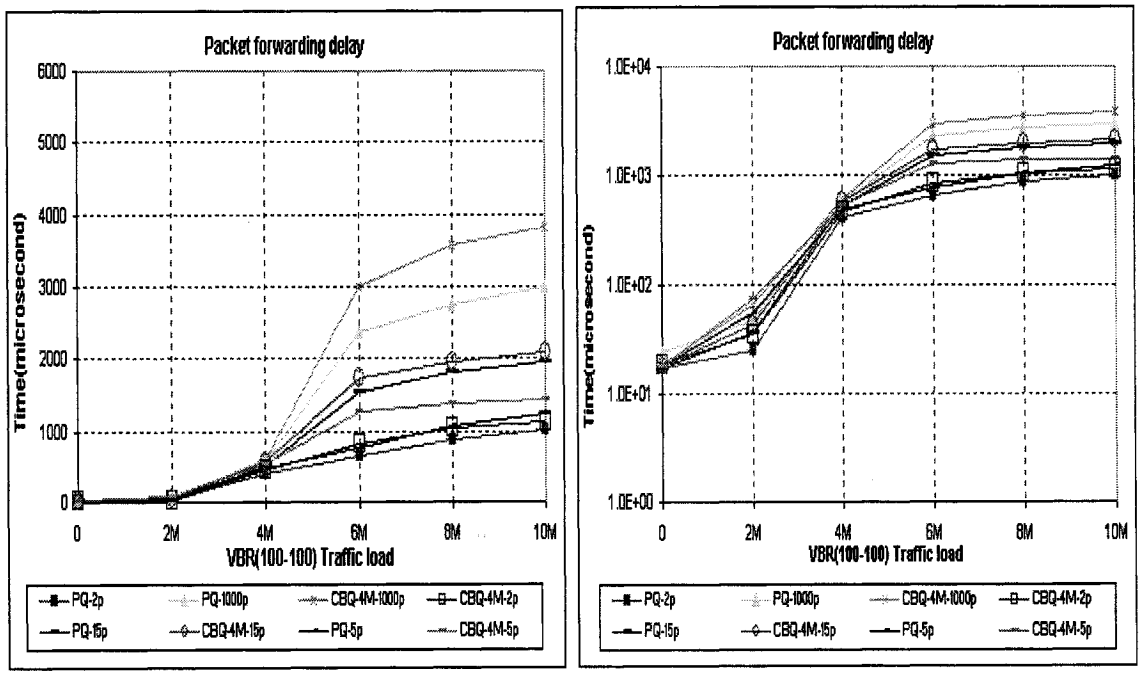
(a) (b)

Figure 5.92 (a) Average Packet Forwarding Delay and (b) its logarithmic graph (50ms-50ms VBR)



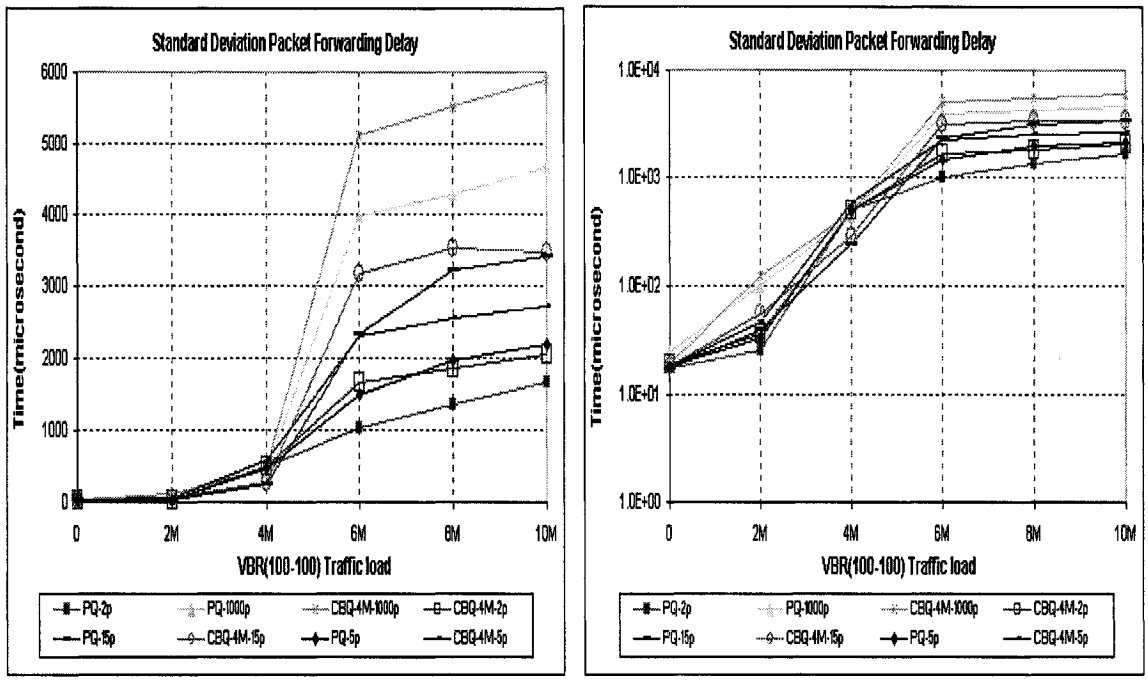
(a) (b)

Figure 5.93 (a) standard deviation of Packet Forwarding Delay and (b) its logarithmic graph (50ms-50ms VBR)



(a) (b)

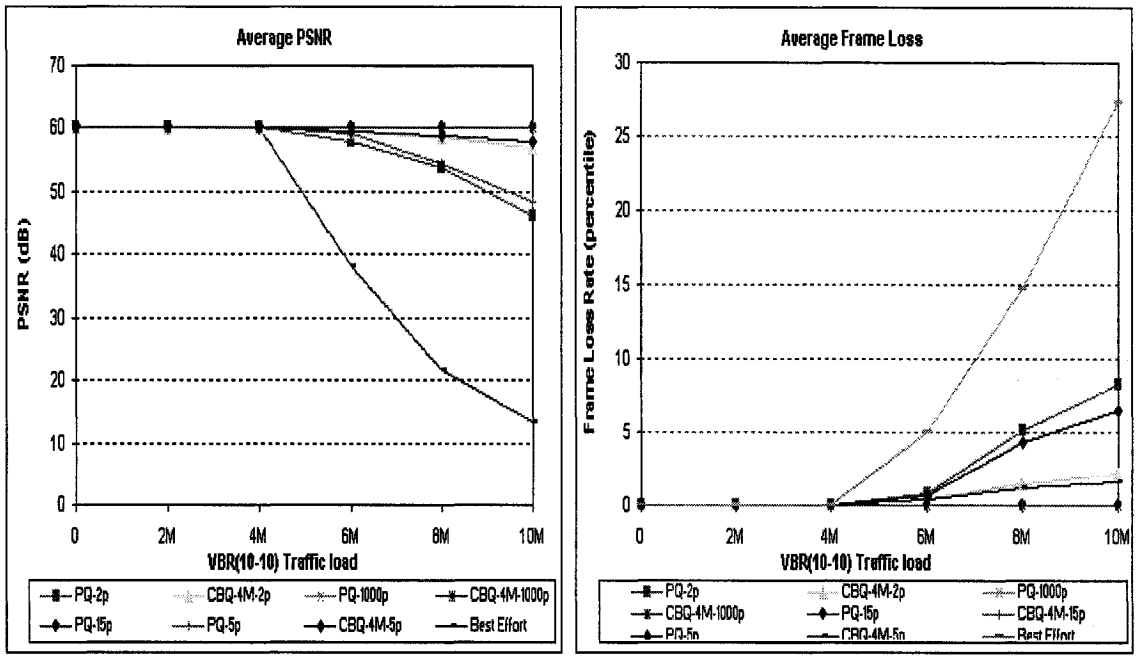
Figure 5.94 (a) Average Packet Forwarding Delay and (b) its logarithmic graph (100ms-100ms VBR)



(a) (b)

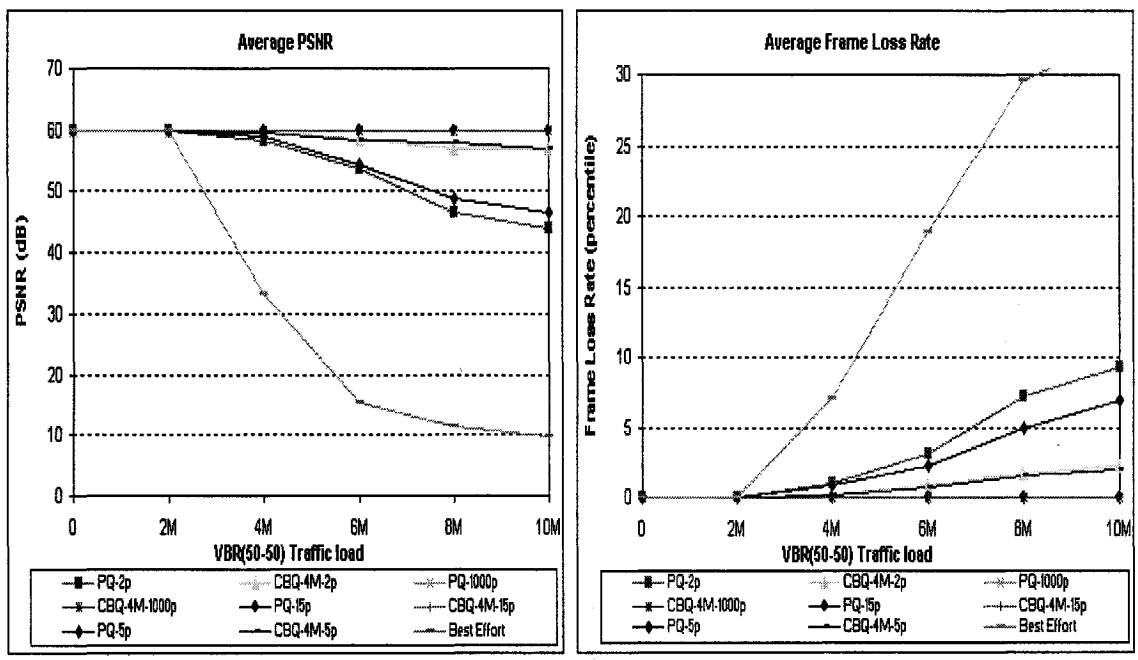
Figure 5.95 (a) standard deviation of Packet Forwarding Delay and (b) its logarithmic graph (100ms-100ms VBR)

From following figures 5.96a, 5.97a and 5.98a, we can see that, DiffServ is able to protect the video over the entire background traffic loading scale. In addition, we observe that CBQ demonstrates superior behavior than PQ, especially for short buffer size. The PSNR graphs show that CBQ keeps the PSNR value larger than 55dB even with 10Mbps, 100ms-100ms VBR when the buffer size is 2 packets, almost 15dB higher than PQ scheme. Comparing with PQ in terms of frame loss rate, we find that the packet loss rate and frame loss rate retain lower value under CBQ, as shown in figure 5.96b, 5.97b and 5.98b. The reason for this behavior is that CBQ is a resource sharing algorithm. It isolates the EF traffic from BE traffic, therefore, the video stream is less affected by the increase in the BE traffic.



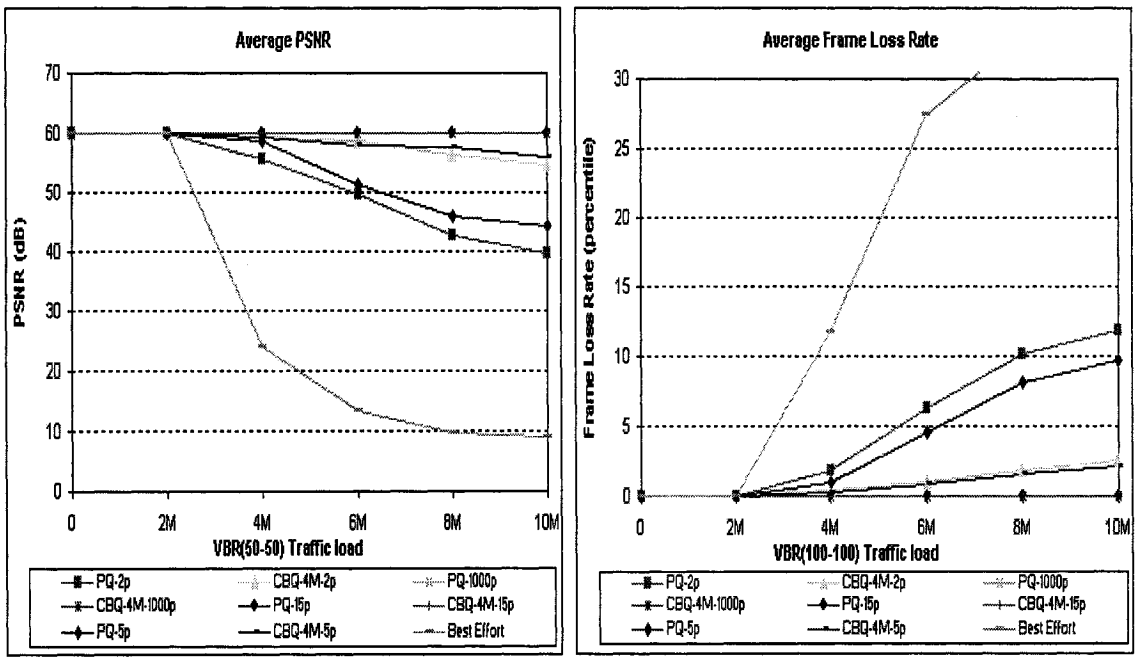
(a) (b)

Figure 5.96 (a) Average PSNR and (b) Average Frame Loss Rate (10ms-10ms VBR)



(a) (b)

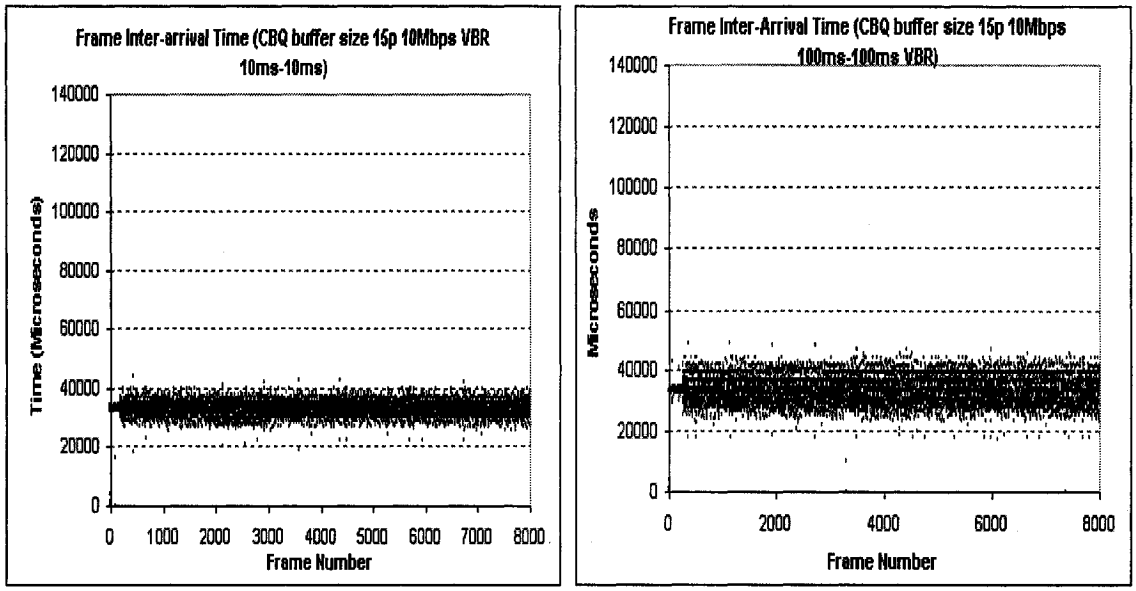
Figure 5.97 (a) Average PSNR and (b) Average Frame Loss Rate (50ms-50ms VBR)



(a) (b)

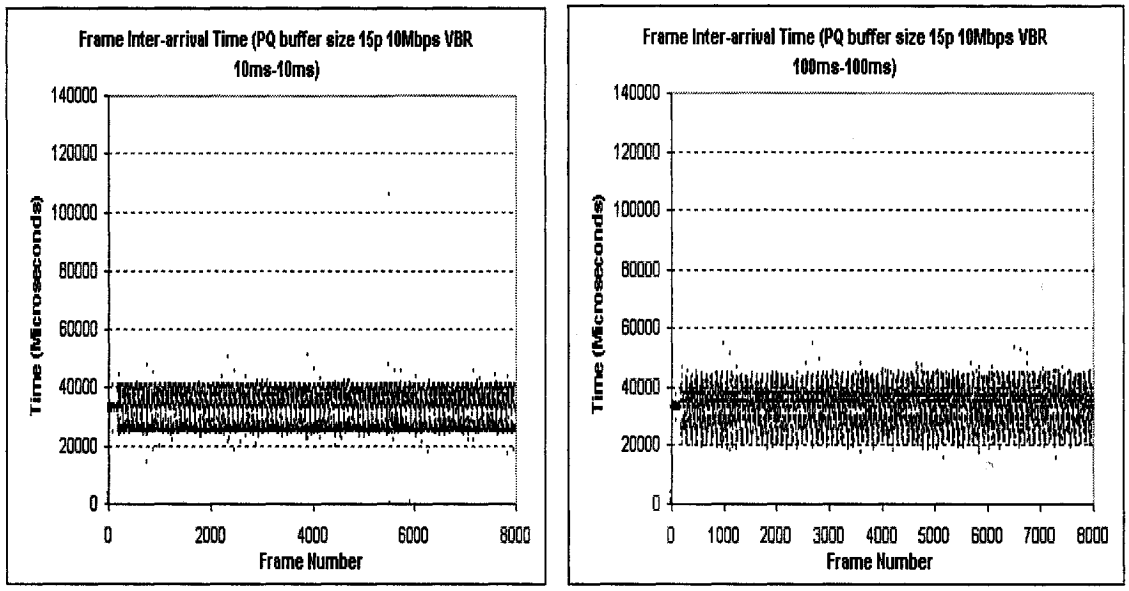
Figure 5.98 (a) Average PSNR and (b) Average Frame Loss Rate (100ms-100ms VBR)

From figures 5.99 to 5.102, we can see the recorded frame inter-arrival time with 10Mbps 10ms-10ms and 100ms-100ms VBR under CBQ and PQ with buffer size 15 and 5 packets. It is evident that for both CBQ and PQ, the recorded inter arrival time forms a belt around 33.2ms. For 100ms-100ms, the belt is wider than 10ms-10ms, this is due to the longer burst periods of the traffic. The improvement is remarkable when we compare these figures with those of best effort, figures 5.87, 5.88. The CBQ shows better performance than PQ because CBQ keeps more packets in the buffer and less packet losses occurring, thus reduces the frame inter-arrival time. The curves of average frame inter-arrival time and its standard deviation are presented in figures 5.103, 5.104 and 5.105. From these figures we can see that because of CBQ's superior behavior, it not only keeps the frame inter-arrival time around 33.2ms but also maintains the standard deviation below 4ms over the entire range of traffic loading. Best effort and PQ experience significant increase of the frame inter-arrival time and its standard deviation with increased traffic loading.



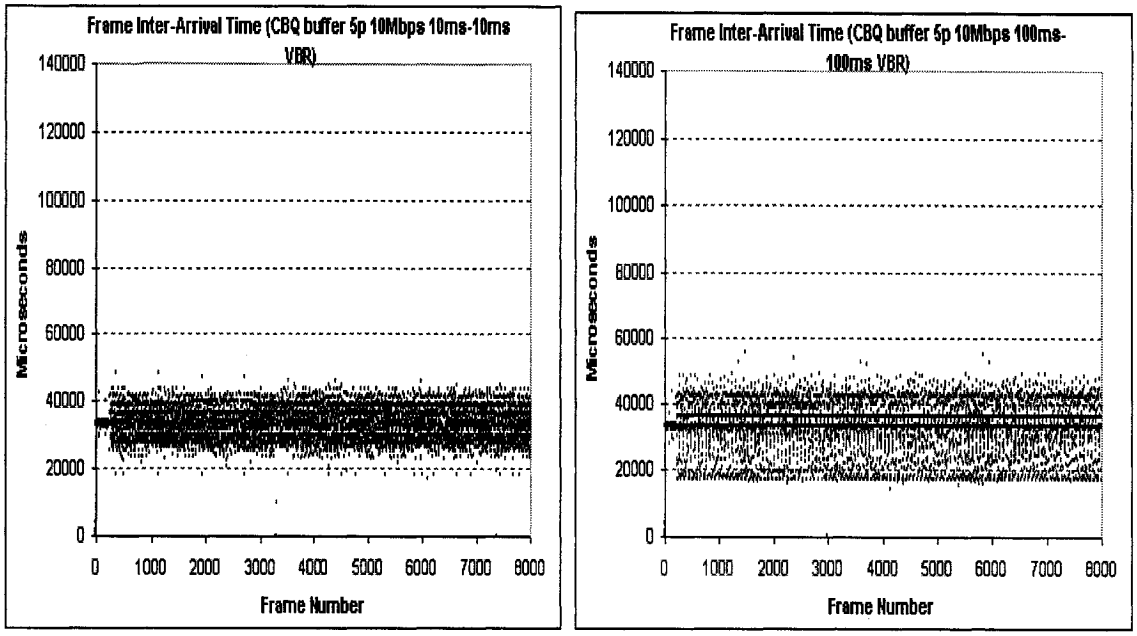
(a) (b)

Figure 5.99 Frame inter-arrival time for CBQ with buffer size 15 packets under 10Mbps background traffic: (a) 10ms-10ms and (b) 100ms-100ms



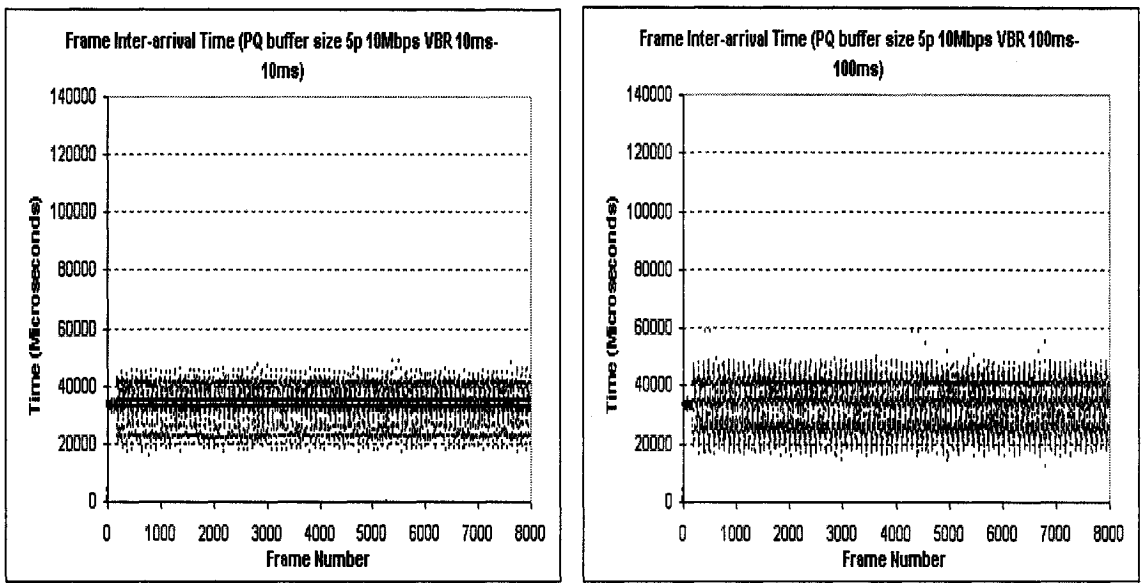
(a) (b)

Figure 5.100 Frame inter-arrival time for PQ with buffer size 15 packets under 10Mbps background traffic: (a) 10ms-10ms and (b) 100ms-100ms



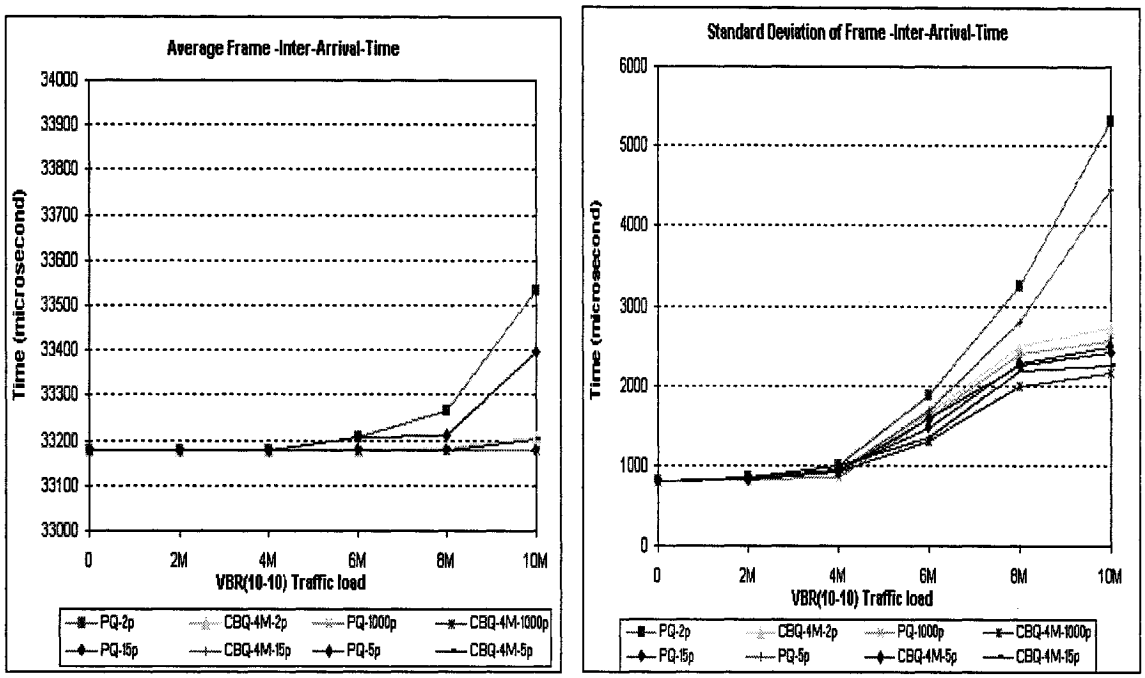
(a) (b)

Figure 5.101 Frame inter-arrival time for CBQ with buffer size 5 packets under 10Mbps background traffic: (a) 10ms-10ms and (b) 100ms-100ms



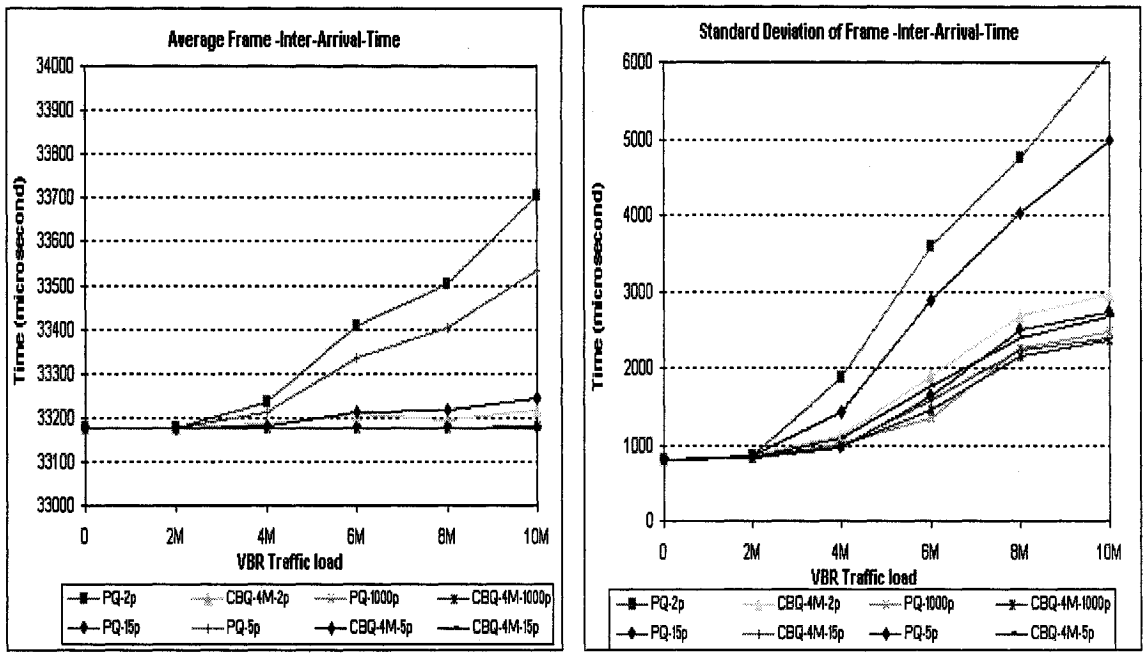
(a) (b)

Figure 5.102 Frame inter-arrival time for PQ with buffer size 5 packets under 10Mbps background traffic: (a) 10ms-10ms and (b) 100ms-100ms



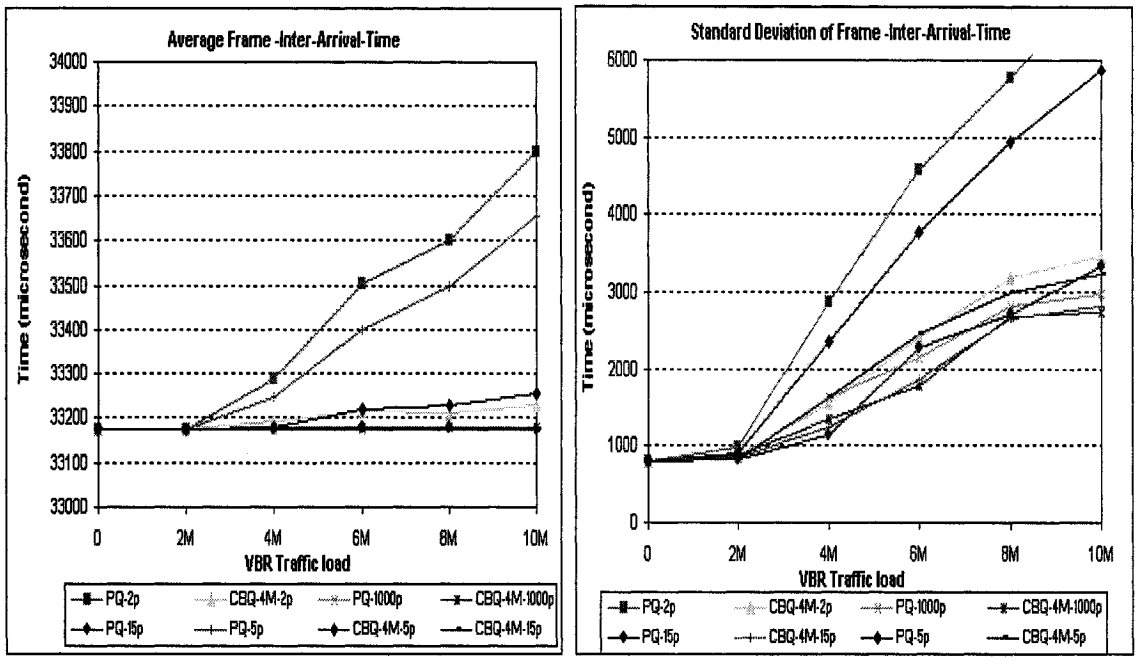
(a) (b)

Figure 5.103 (a) Average frame inter-arrival time and (b) its standard deviation (10ms - 10ms VBR)



(a) (b)

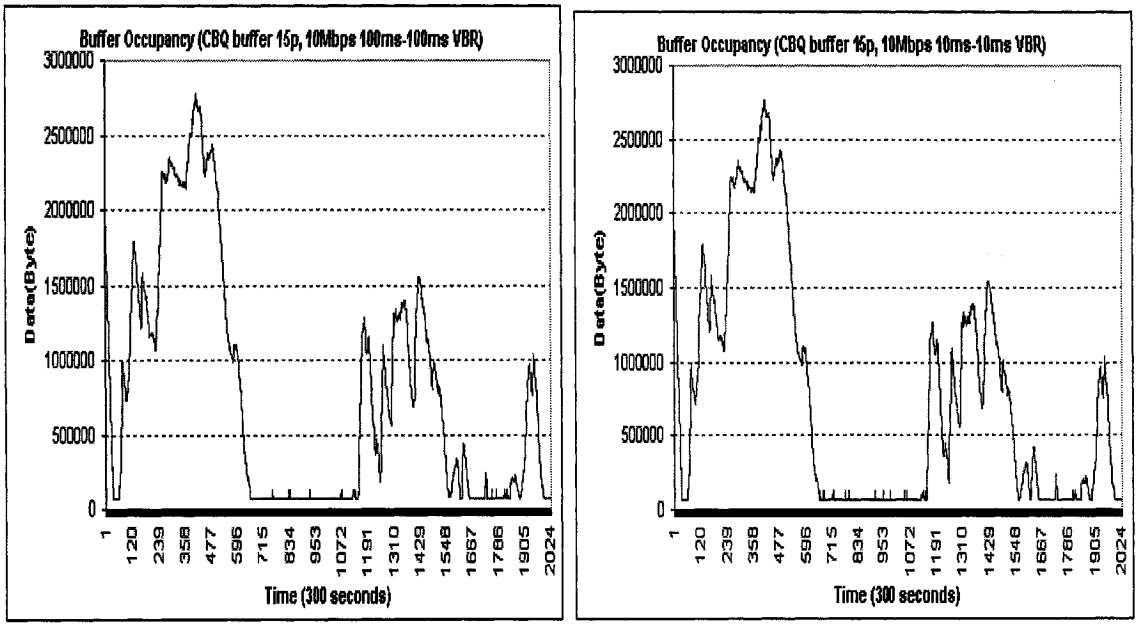
Figure 5.104 (a) Average frame inter-arrival time and (b) its standard deviation (50ms-50ms VBR)



(a) (b)

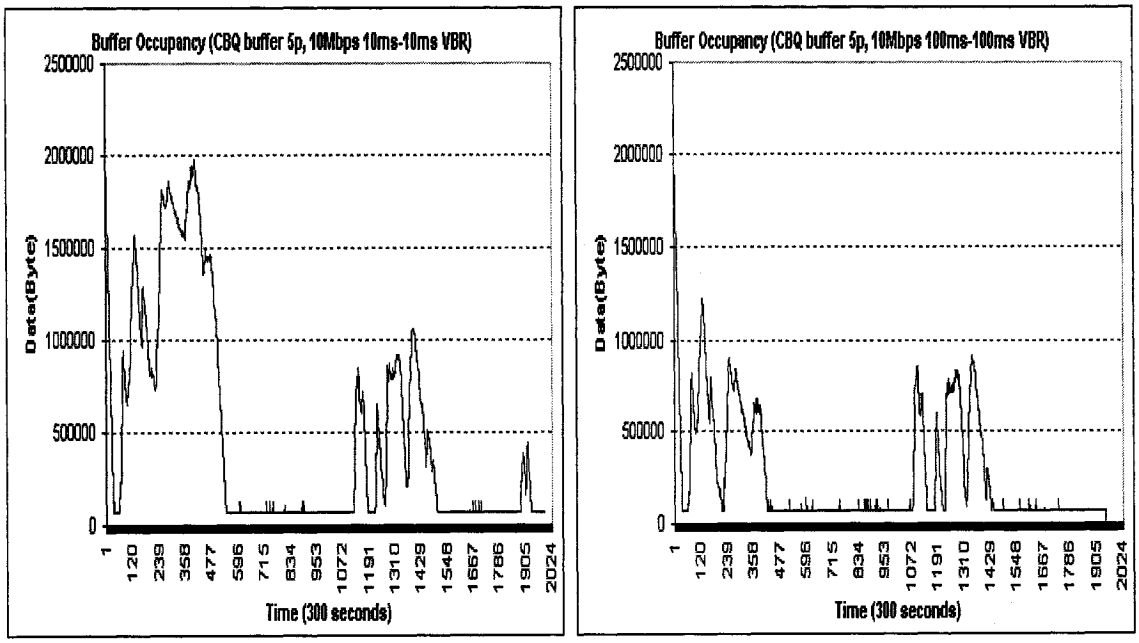
Figure 5.105 (a) Average frame inter-arrival time and (b) its standard deviation (100ms-100ms VBR)

The sustainability of connection is also well protected by the DiffServ's CBQ and PQ schemes with suitable EF buffer size (15 packets or more). Curves in figure 5.106a and 5.106b present the buffer occupancy with CBQ buffer size 15 packets when 10Mbps 100ms-100ms and 10ms-10ms VBR are applied. It is evident that the graph shape in figures 5.106a and 5.106b is identical to that in figure 5.44, our benchmark with 0% background traffic loading. This means that all video content is received by the client despite the high bursty traffic loading of 10Mbps. Even for the short buffer size case, the performance has been highly improved as shown in figure 5.107a, 5.107b when comparing with figure 5.49. The average buffer occupancy graphs are given in figures 5.108, 5.109 and 5.110. Under DiffServ, the decoder never enters into the starvation state even with high traffic loading and short buffer size. The sustainability of connection is maintained. In addition, we find that CBQ always maintain higher buffer occupancy than PQ under small buffer size allocation at the core router.



(a) (b)

Figure 5.106 Buffer occupancy under CBQ with buffer size of 15 packets: (a) 10Mbps 100ms-100ms VBR and (b) 10ms-10ms VBR



(a) (b)

Figure 5.107 Buffer occupancy under CBQ with buffer size of 5 packets: (a) 10Mbps 100ms-100ms VBR and (b) 10ms-10ms VBR

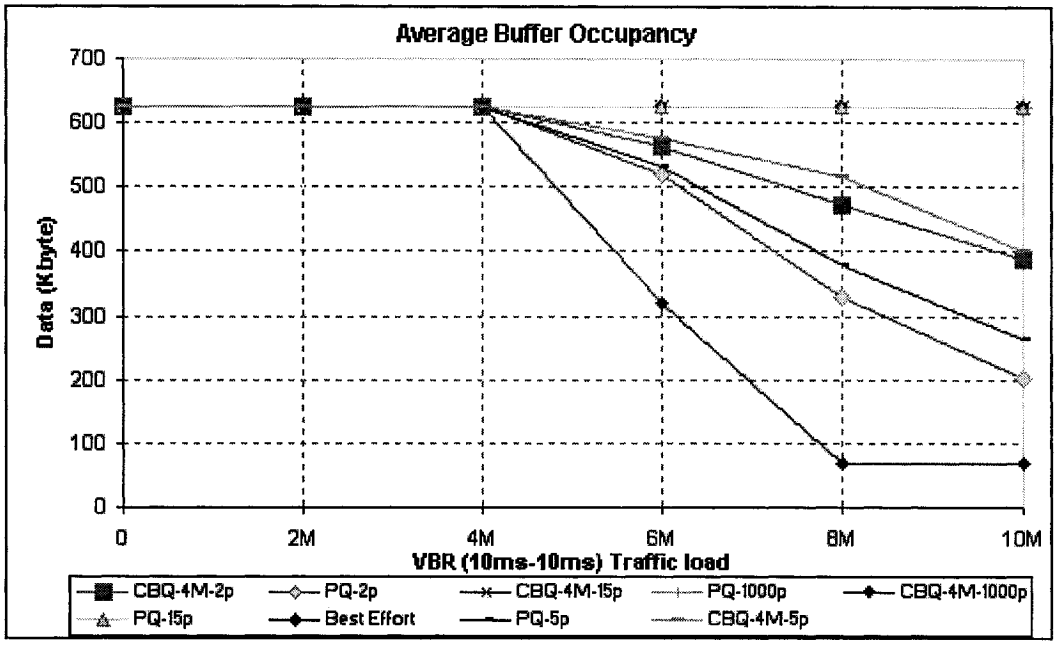


Figure 5.108 Average Buffer Occupancy (VBR 10ms-10ms)

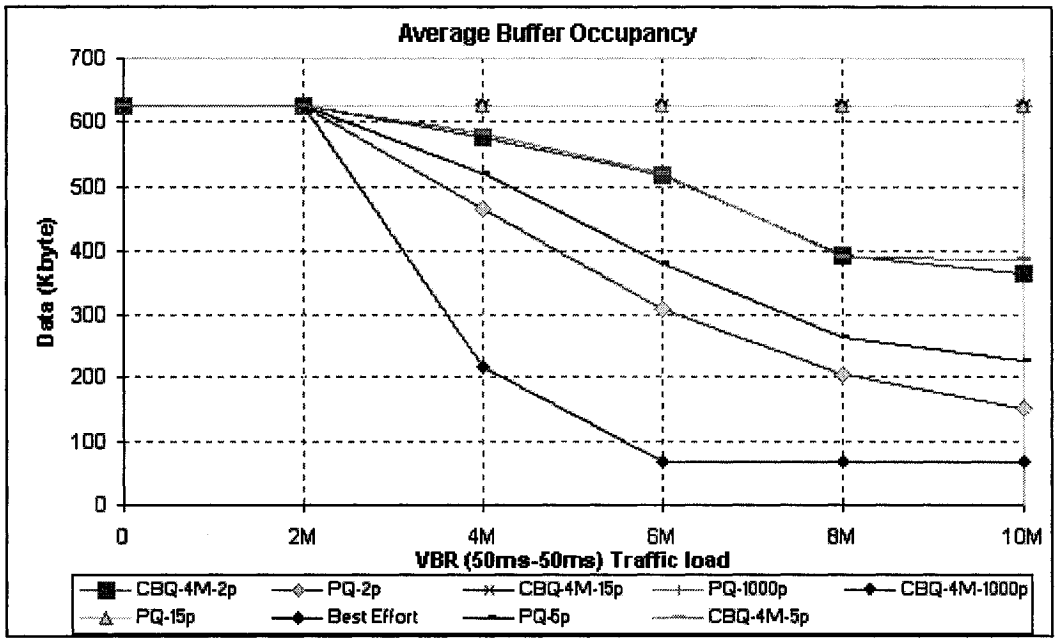


Figure 5.109 Average Buffer Occupancy (VBR 50ms-50ms)

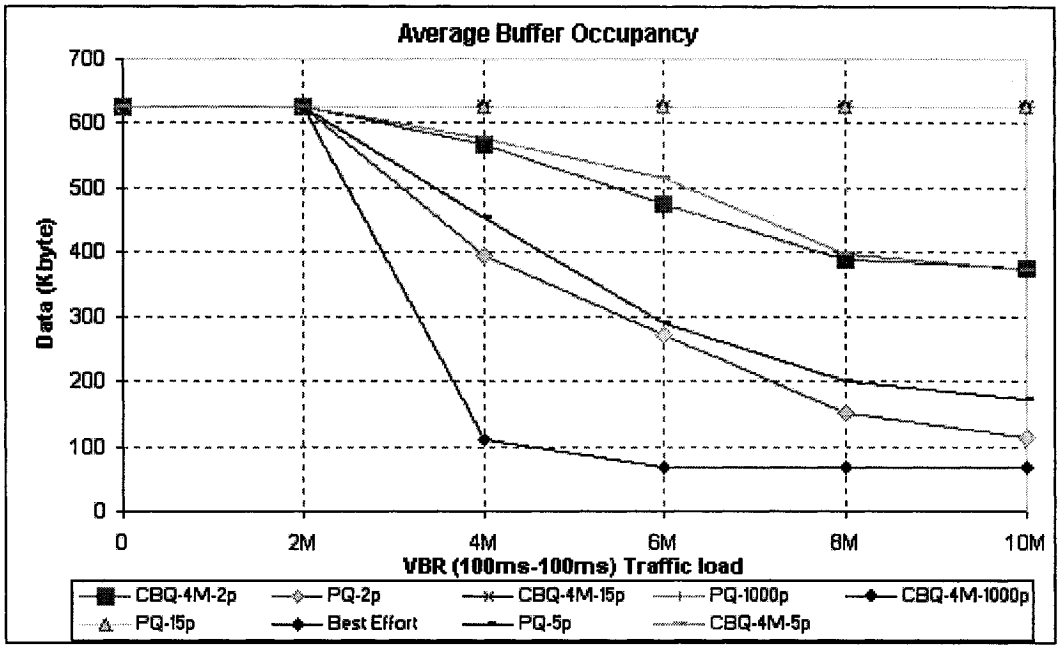


Figure 5.110 Average Buffer Occupancy (VBR 100ms-100ms)

In summary, we find that best effort cannot support video with desirable quality in the presence of competing traffic. However, Differentiated Service resolved this problem.

By analyzing the test results, we conclude that CBQ is a better scheme than PQ under DiffServ for the support of video applications. Because the performance of PQ is considerably affected by background traffic packet size and buffer size, its video supporting ability is not as expected in a real network. It will show considerable venerability as the traffic loading will be changing with time, more difficult for the service provider to maintain certain level of guarantees. Therefore, we believe that the CBQ under DiffServ is the best choice for providing the desirable QoS guarantee.

5.3 Distributed Interactive Virtual Environment (DIVE) Measurement & Analysis

Based our earlier research on networked DIVE applications, the authors of [GAL99] developed a DIVE traffic model and implemented a traffic generator, reproducing traffic according to the statistics of this model. The generator is used in our test-bed to emulate the DIVE traffic, thus allowing us to run the lengthy experiments, which would be impossible should actual human operators had to be used for the purpose of operating the DIVE application in order to generate traffic. Figure 5.111 shows a trace of DIVE traffic generated through our traffic generator. The average rate for this trace is in the range of 5Kbit/s.

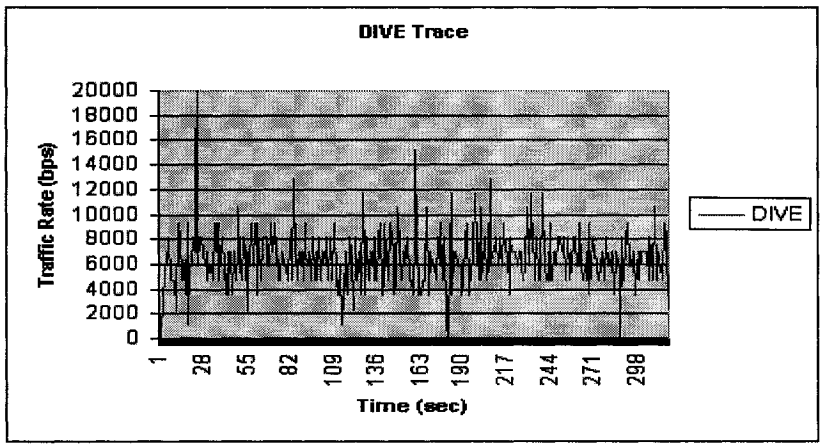


Figure 5.111 DIVE traffic trace

In the DiffServ network, DIVE traffic is served as EF class, with packet forwarding policies of Priority Queuing (PQ) or Class-Based Queuing (CBQ) set at the core router. A First-In-First-Out (FIFO) queuing algorithm is used for the packets of the various EF classified traffic streams, entering the EF queue of the core routers.

When deploying DIVE applications through Internet, DIVE traffic will have to share resources with the traffic of other real-time applications (in our work, we use video traffic). As mentioned earlier, a FIFO approach is followed in EF class for all arriving packets. When these competing sources produce persistently bursty traffic (as is the case of video) they might consume all available resources within the periods of high

burstiness, thus forcing the DIVE traffic to experience losses and larger packet delays. This could severely impair the DIVE application. In [HYU01], the authors presented test results to explore how the DiffServ mechanism supports DIVE. The trade-off still exists: long buffer can hold packets but delay and system expense increases, short buffer reduces delay and saves system resources but packet loss happens. In [HYU01], the authors proposed the EF2 QoS architecture for solving the problem: It consisted of splitting the buffer space of EF class in two buffers. One was allocated to DIVE traffic, the second to other types of real time traffic (e.g. video, voice etc.). Analysis indicated that the EF2 scheme provided superior performance when compared to a FIFO policy at EF queue (single queue, aggregating all EF traffic streams) when the suitable buffer size is chosen. In my work, a different packet forwarding policy at the EF class is proposed and the results of comparison are presented.

5.3.1 Virtual Queue Algorithm (VQ)

The packet forwarding policy followed by the core routers in the DiffServ architecture determines the performance of the DIVE application. As mentioned earlier, in the regular DiffServ architecture, packets are generally served on a FIFO basis. In our case, DIVE and Video packets will be served in sequence, according to their arrival time in the queue. When the queue is filled because of bursty video traffic, it is inevitable that DIVE traffic packets will be dropped or experience longer queuing delays. Figure 5.112 demonstrates the process.

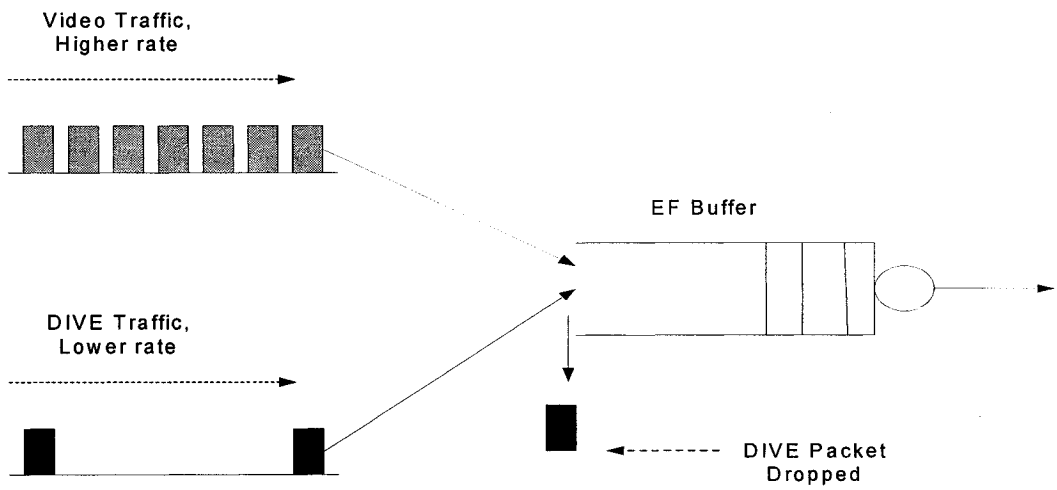


Figure 5.112 DIVE and Video packets in EF buffer in Core router

We propose an alternative queuing approach: Virtual Queue (VQ). When Core router receives EF traffic, the first step is to decide the packet type. An arriving DIVE packet is placed in front of the video packet that is ahead of all other video packets in the queue. Therefore we place DIVE traffic packets at the head of the queue, while maintaining the FIFO policy for the remaining EF traffic. VQ gives the priority to the DIVE packets as they pass through the core routers. Figure 5.113 shows the working process of VQ.

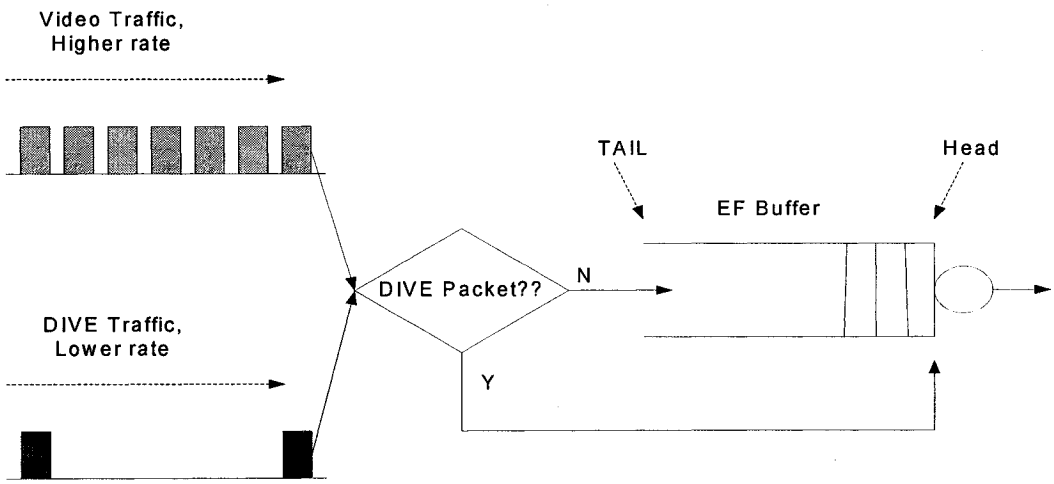


Figure 5.113 DIVE and Video Packets in EF Buffer under VQ in Core router

In order to implement the VQ discipline in the DiffServ core router, the DiffServ edge router needs to mark the DIVE and Video traffic further to distinguish DIVE traffic from other EF traffic. We can use the TCP option byte or an unused TOS byte inside the packet header for marking. Figure 5.114 shows the byte we used for marking DIVE and Video packets. To our work, we use these bytes and mark DIVE and Video packet as 0xb9 and 0xb8. In this case, their DSCP value still is 0x2e (EF class).

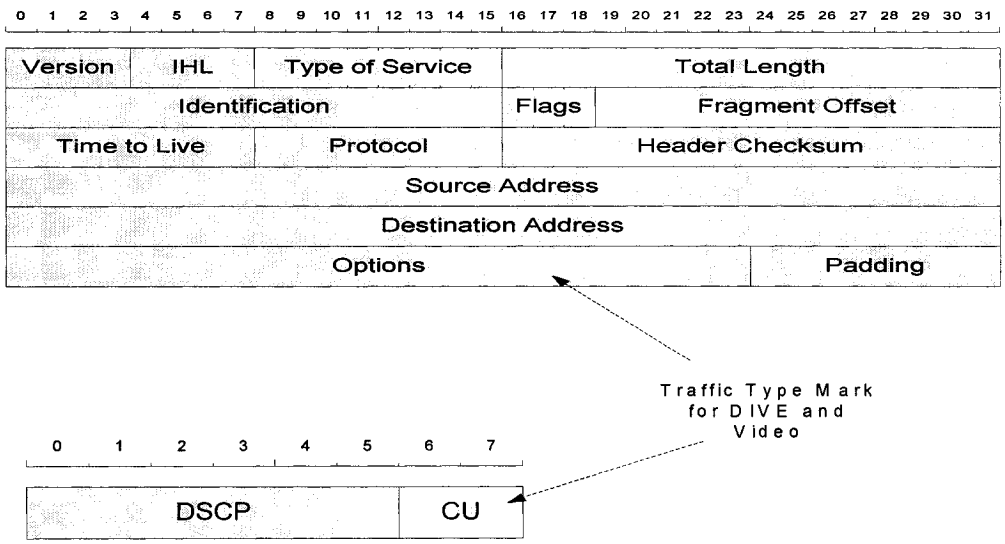


Figure 5.114 Bytes in packet header used for marking DIVE and VIDEO

We implemented the VQ algorithm in our experimental testbed and contacted experimental performance evaluation. The Linux kernel 2.4.3 has been used. We modified the Linux router traffic control (scheduling discipline) and implemented the iproute2+tc traffic control package [KUZ98] as in Figure 5.115. Queuing discipline FIFO, Random Early Drop (RED), PQ, CBQ have been built inside the kernel.

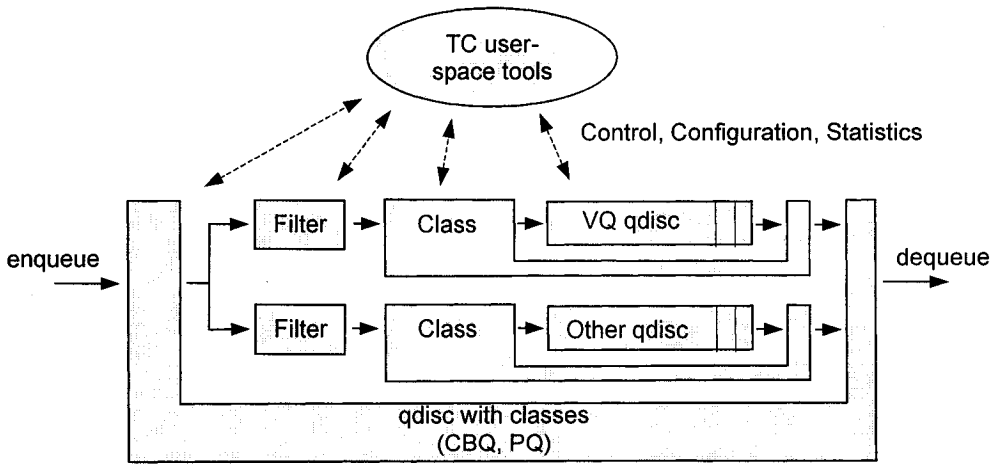


Figure 5.115 VQ implementation under TC

In our test-bed, DIVE application and video application (MPEG2) traffic coexist together to emulate the "real" network by passing the DiffServ-enabled network. UDP is used as

the transport layer protocol. Since both applications are sensitive to delays, there is no reason of re-transmitting lost packets. Video is produced through the MPEG2 compressed video-on-demand (VoD) application described earlier. As mentioned earlier, for this particular stream, the observed average video transmission rate is close to 4 Mbps, while the observed maximum and minimum rates are 4.8 Mbps and 3.3 Mbps. The CBR background traffic is used to produce the link congestion at the output of the core router.

5.3.2 Performed Tests and Results Analysis

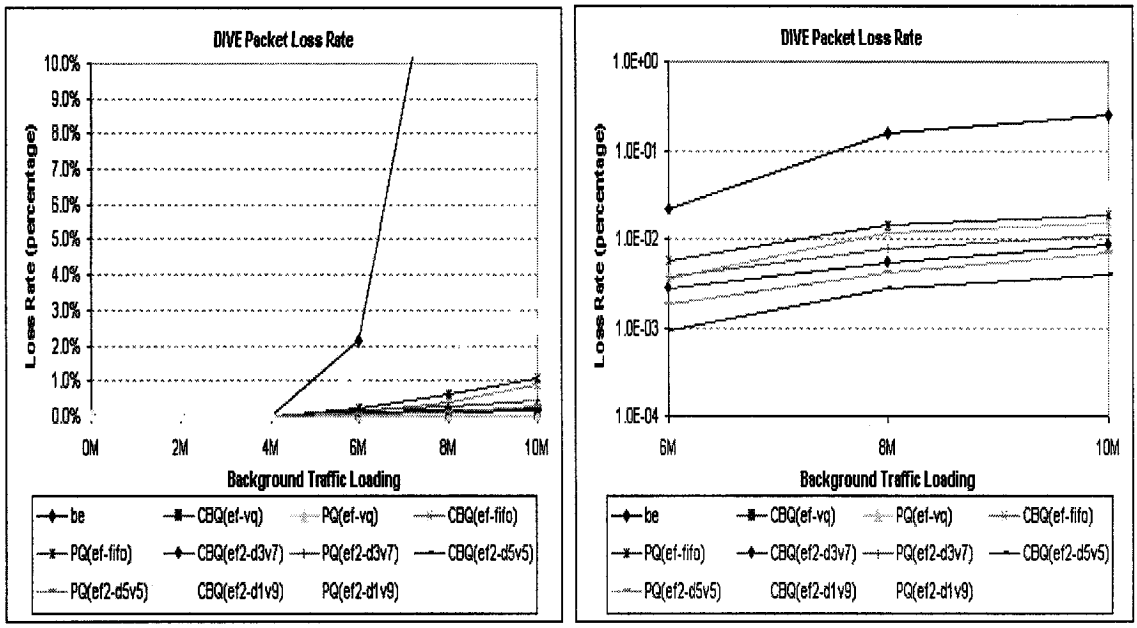
A comprehensive set of experiments has been conducted and the detailed statistical information has been collected and processed. We present results concerning the packet loss rate of the DIVE application, as well as the packet forwarding delay, when the video application and the background traffic co-exist in the network. Considering the worst case, we let the background traffic packet length to be as long as 1500 bytes. We evaluated the following configurations: 1). Best effort network with buffer size 105 packets. 2). DiffServ network with PQ, with the DIVE and video traffic streams placed in the same buffer at EF using a FIFO queue discipline, EF buffer size 10 packets; background traffic is placed at the best effort class, PQ(ef-fifo). 3). DiffServ network with CBQ, with the DIVE and video traffic streams placed in the same buffer at EF using FIFO queue discipline and EF buffer size 10 packets; background traffic is placed at the best effort class, CBQ(ef-fifo). 4). DiffServ network with PQ, with the DIVE and video traffic streams placed in the same buffer at EF using VQ queue discipline and buffer size 10 packets; background traffic is placed at the best effort class, PQ(ef-vq). 5). DiffServ network with CBQ, with the DIVE and video traffic streams placed in the same buffer at EF using VQ queue discipline, buffer size 10 packets; background traffic placed at the best effort class, CBQ(ef-vq). 6). DiffServ network with PQ EF2, with the DIVE and video traffic streams assigned to the EF class but placed in different queues, with DIVE buffer with size X={1,3,5} and Video buffer size Y=10-X. 7). DiffServ network with CBQ EF2, with the DIVE and video traffic streams assigned to the EF class but placed in different queues with DIVE buffer size X={1,3,5} and Video buffer size Y=10-X. For

CBQ, the resource allocation to DIVE is 7Kbps, whereas the resource allocation to video is 4Mbps. Borrowing is not allowed.

Figure 5.116 displays the rate of packet losses experienced by the DIVE traffic stream at the core router for different values of background traffic (CBR). Since the packet losses experienced by DIVE application should be very low, it is evident that a best effort network is not capable of supporting such applications. On the contrary, the DiffServ mechanism is performing quite well. From the following figures, we can make a comparison between the various DiffServ schemes.

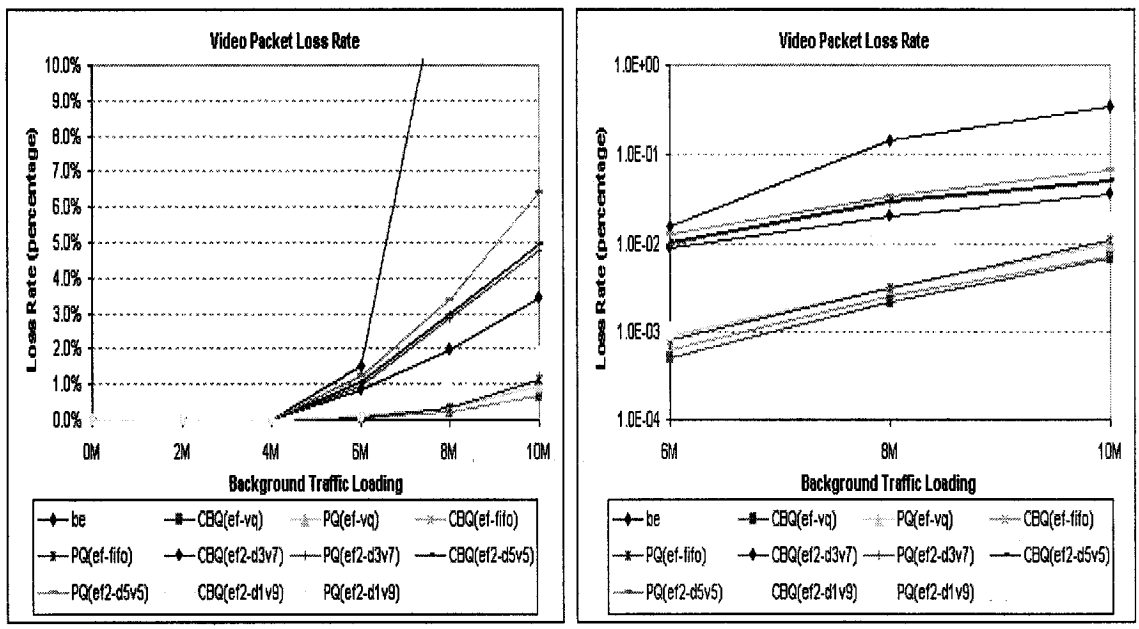
First, we compare VQ with FIFO under Priority Queuing (PQ) and Class Based Queuing (CBQ). Figures 5.116 displays the percentage of packet losses experienced under VQ and FIFO disciplines (applied at EF) versus background traffic loading, when PQ or CBQ are used as traffic forwarding policies between the EF and BE buffers. Figure 5.117 provides the same information (percentage of packet losses versus background traffic loading), but for the video packets this time. From figure 5.116 and figure 5.117, we can see that when FIFO is used at the EF queue, while PQ is used between EF and BE queues, the loss rate of DIVE packets is around 0.5% for 80% background traffic loading (8 Mbps), and close to 1.2% for 100% background traffic loading (PQ(ef-fifo)). For FIFO under CBQ, the loss rate of DIVE packets is 0.35% for 80% background traffic loading and 1.2% for 100% background traffic loading (CBQ(ef-fifo)). However, by inspecting these figures, it is quite evident that when VQ is used, the losses of DIVE packets diminish, being very close to zero regardless of the background traffic loading levels. This holds for both, the PQ and CBQ configurations. Inspecting Figure 5.117, we realize that the loss rate of video packets remains almost the same regardless of what you use FIFO or VQ policy. Again, use of CBQ rather than PQ provides a better performance under heavy loading. Figure 5.118 provides the average forwarding delay experienced at the core router by DIVE packets, when PQ or CBQ is used as the packet forwarding policy between EF and BE queues. It is evident that VQ provides slightly better performance at higher loading levels.

Second, we wish to see how VQ policy compares to EF2. For this purpose we ran a set of experiments for the following configurations (system resource total buffer size 10 packets.): a) Use of VQ at EF with EF buffer size equal to 10 packets (Vqbuffer10). b) Use of EF2 at EF with 5 packets capacity allocated to DIVE queue, 5 packets to the second EF queue (EF2v5d5). c) Use of EF2 at EF with 3 packets capacity allocated to DIVE queue, 7 packets to the second EF queue (EF2v7d3). d) Use of EF2 at EF with 1 packet capacity allocated to DIVE queue 9 packets to the second EF queue (EF2v9d1). The percentage of DIVE packet losses is shown in Figure 5.116. The results correspond to CBR background traffic loading and when a PQ or a CBQ policy is used between BE and EF classes. By inspecting Figure 5.116, For EF2, the DIVE packet losses rate reaches 0.3% under PQ and 0.18% under CBQ with DIVE buffer size 5, however, if DIVE buffer is too small DIVE in EF2, the packet losses increase dramatically. We can see that for buffer size 1 packet, the packet loss rate reaches 3.14% under PQ and 2.77% under CBQ. The superiority of VQ when compared to the EF2 architecture becomes evident because of its low packet loss properties. Figure 5.116 and 5.121 display the video packet loss rate and throughput of video, measured at the output of the core router for the VQ and EF2 configurations described above. It is evident that VQ, not only assists the DIVE traffic, but also reduces the video packet loss rate and improves the throughput of video traffic (consequently, its performance as well, since higher throughput reduces the queue buildup and the consequential video packet losses).



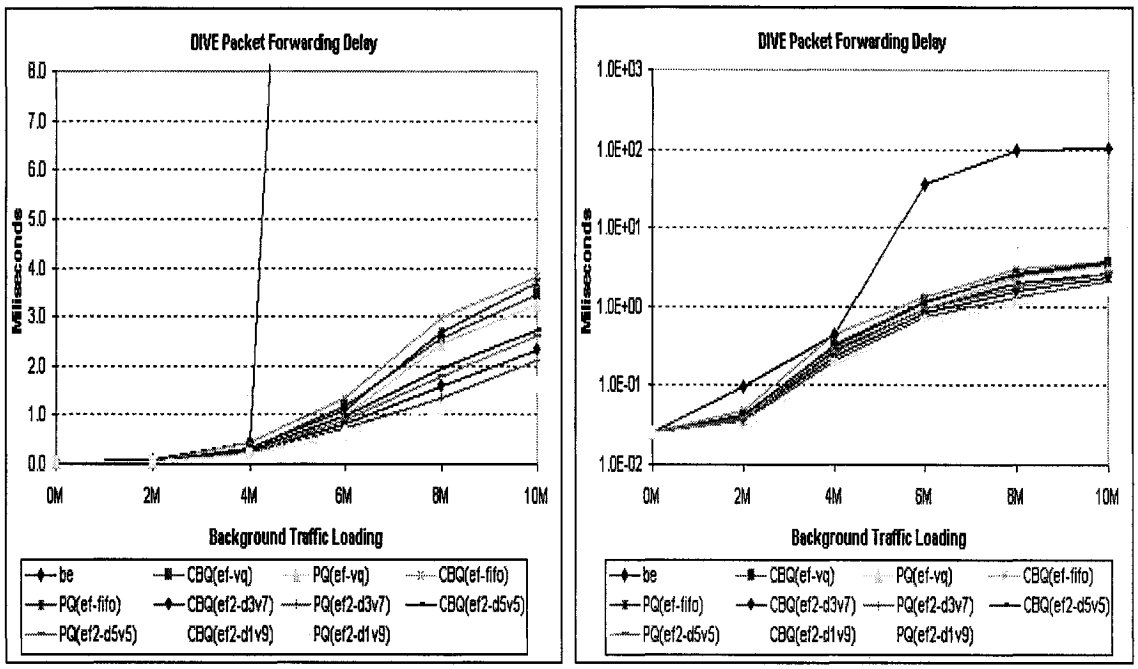
(a) (b)

Figure 5.116 Average Packet Loss Rate experienced by DIVE packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs



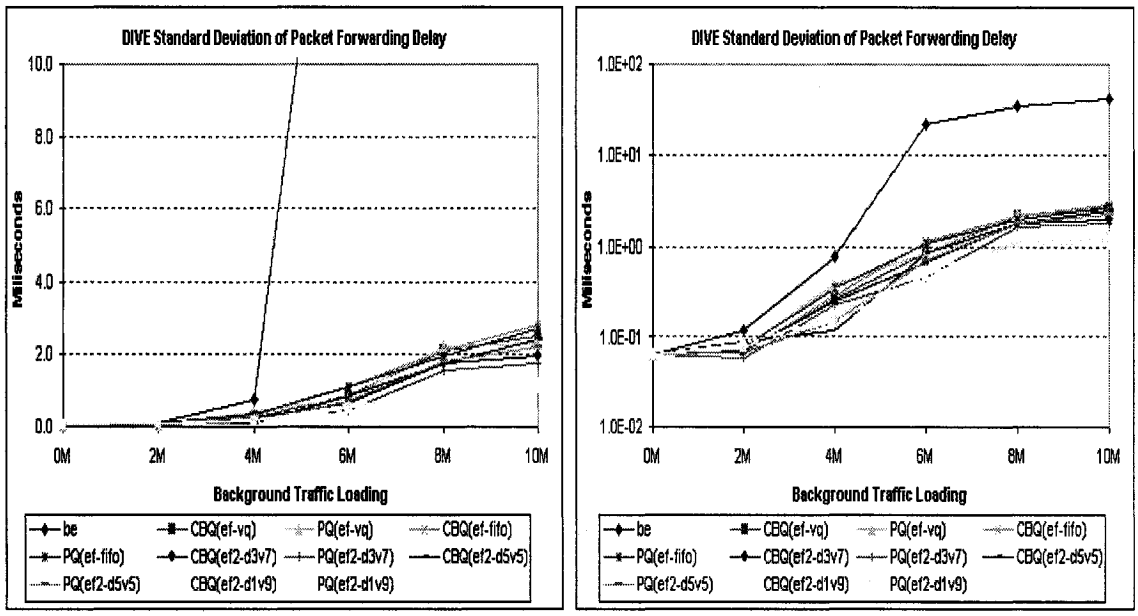
(a) (b)

Figure 5.117 Average Packet Loss Rate experienced by Video packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs



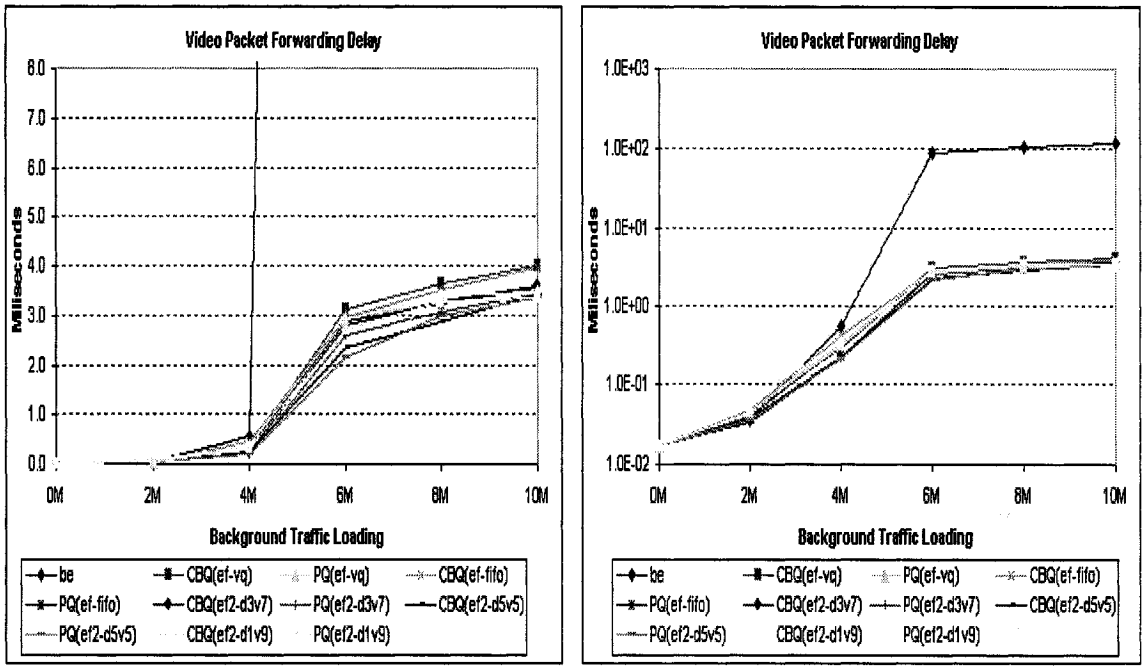
(a) (b)

Figure 5.118 Average Packet Forwarding Delay experienced by DIVE packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs



(a) (b)

Figure 5.119 Standard deviation of average Packet Forwarding Delay experienced by DIVE packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs.



(a) (b)

Figure 5.120 Average Packet Forwarding Delay experienced by video packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs

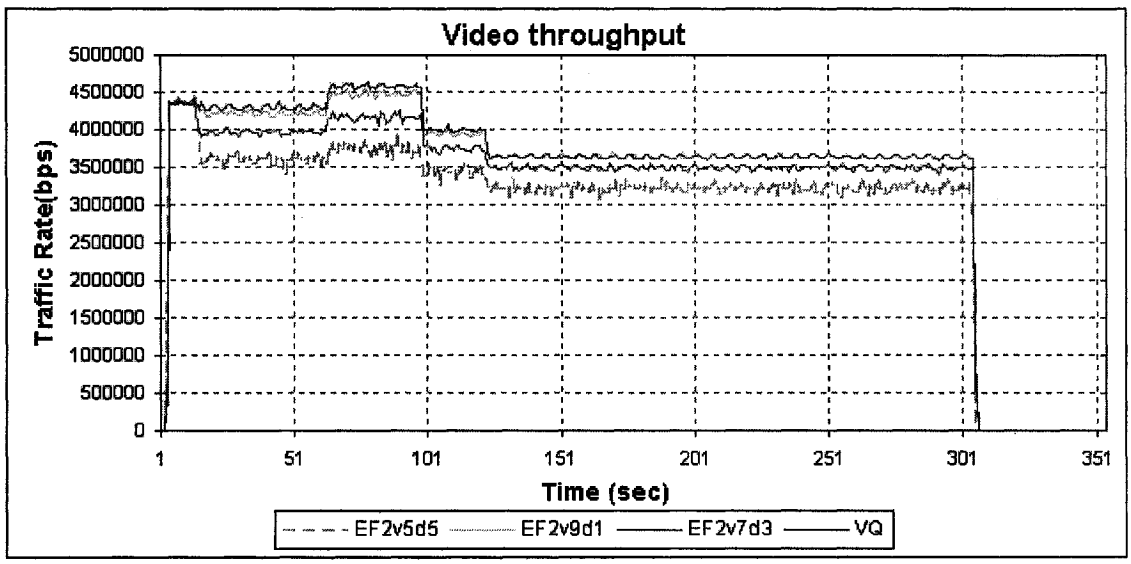
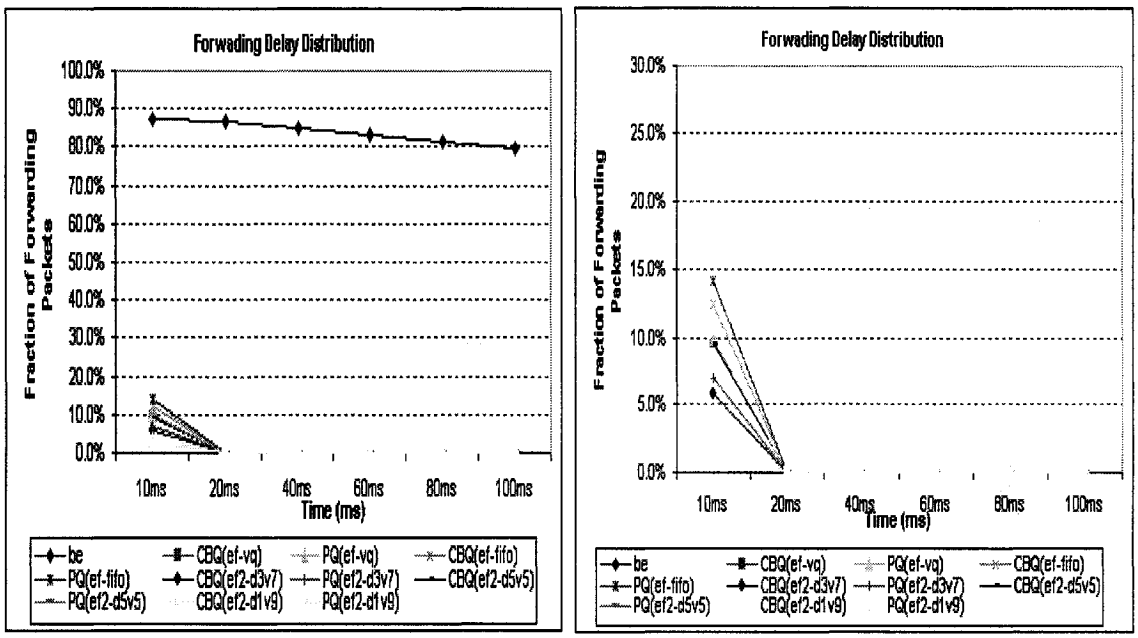


Figure 5.121 Video traffic throughput measured at the output of the core router. (10Mbps CBR background traffic loading)

As mentioned earlier, DIVE applications are highly sensitive to end-to-end delay. Should a packet be delivered 100 msec after generation, it is considered outdated. Figure 5.176

displays the fractions of DIVE packets exceeding certain value of forwarding delay. It is important that per router forwarding delay must be kept at minimal in order to allow for the deployment of DIVE applications over wide geographic areas. It is quite evident that best effort is performing very badly, while the DiffServ configurations are performing well. The fraction of packets experiencing forwarding delay higher than 10 msec is very small for all DiffServ scenarios. In addition, we find CBQ performance is better than PQ. Considering all these performance parameters, CBQ with VQ should be a good choice for deploying DIVE applications over large networks.

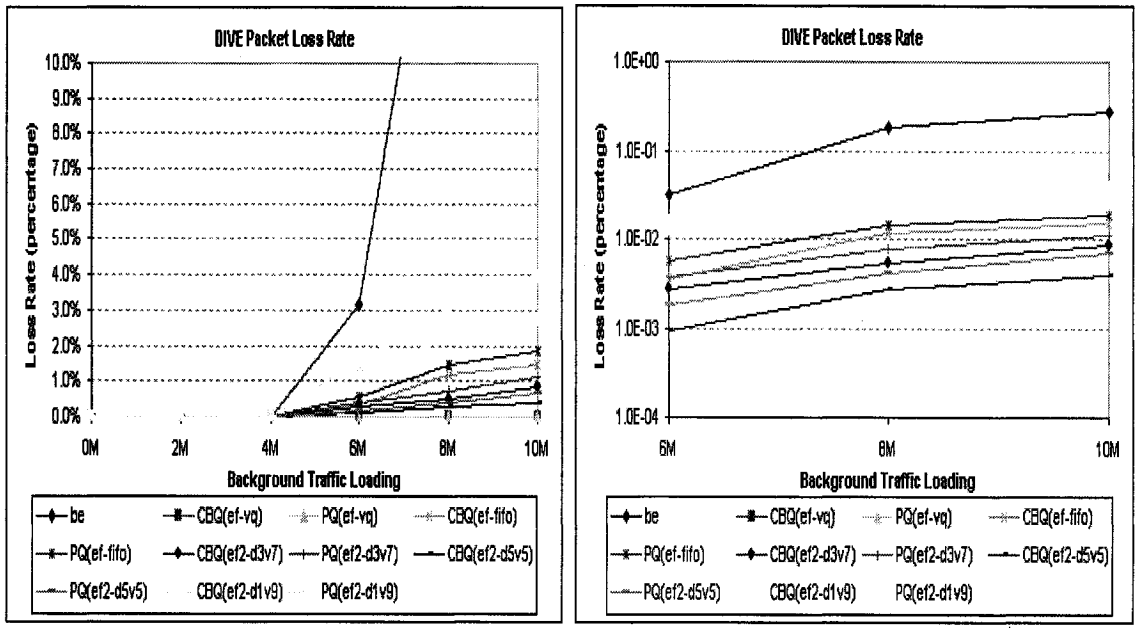


(a) (b)

Figure 5.122 Fraction of DIVE forwarded packets whose forwarding delay exceeds certain size

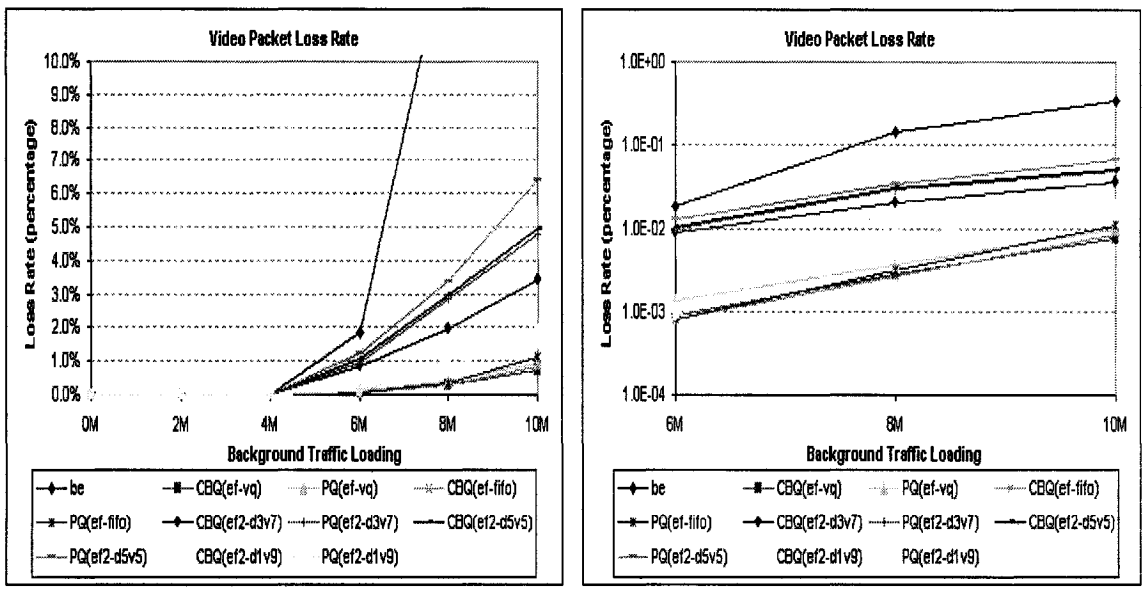
In order to investigate the performance of multiple DIVE applications over DiffServ networks, we also contacted a set of experiments for 20 streams of DIVE applications. The buffer size for EF is still 10 packets for comparison. Figures 5. 123 and 5.124 display the DIVE packet loss rate and video packet loss rate, it is evident that DIVE packet loss rate increases, however, the change is small. For VQ, the packet loss remains at zero level. We also see a slight increase of the loss rate of video packets because of the

aggregated DIVE traffic affection. The packet forwarding delay and the fraction of packets experiencing forwarding delay shown in Figure 5.125 and 5.127 increase slightly.



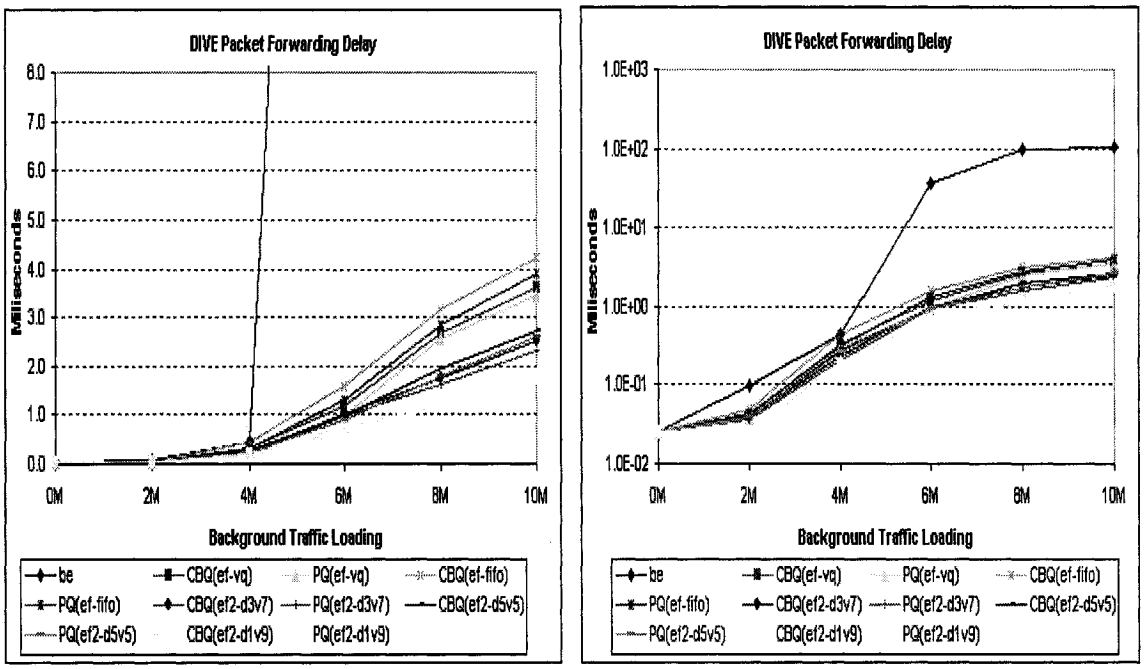
(a) (b)

Figure 5.123 Average Packet Loss Rate experienced by DIVE packets (20 streams) at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs



(a) (b)

Figure 5.124 Average Packet Loss Rate experienced by Video packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs



(a) (b)

Figure 5.125 Average Packet Forwarding Delay experienced by DIVE packets (20 streams) at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs

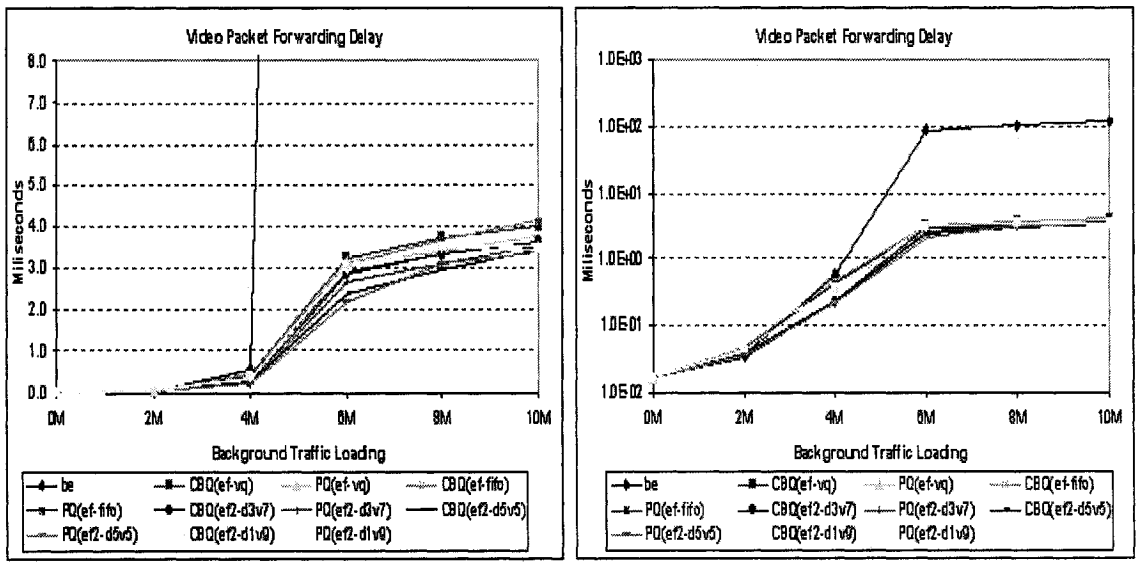
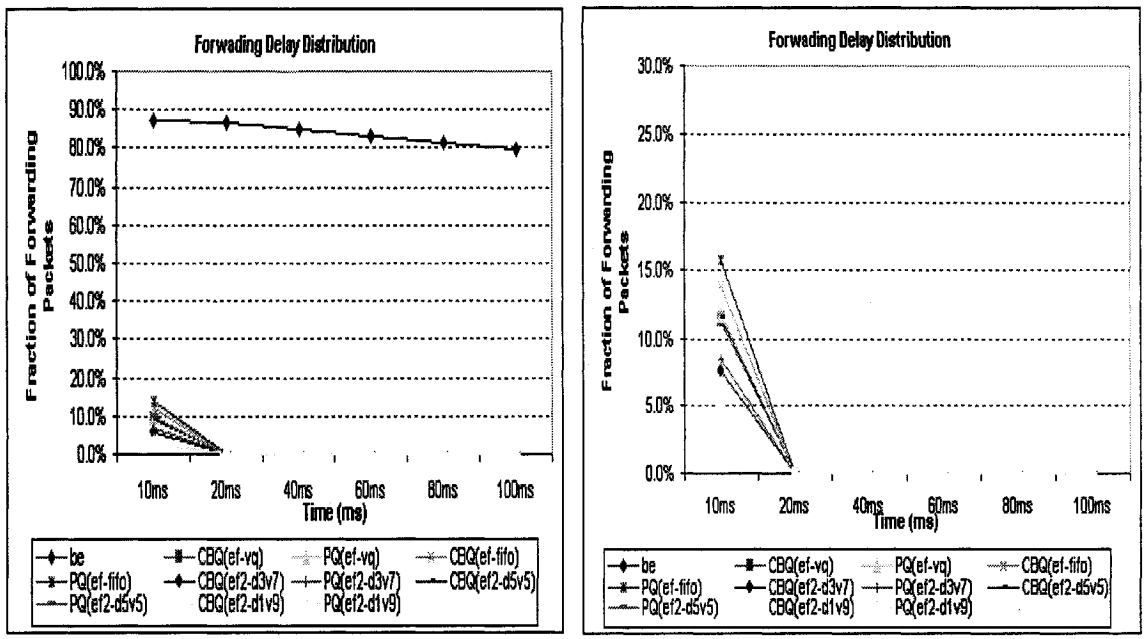


Figure 5.126 Average Packet Forwarding Delay experienced by Video packets at the core router versus traffic load: (a) linear and (b) logarithmic scale graphs



(a) (b)

Figure 5.127 Fraction of DIVE (20 streams) forwarded packets whose forwarding delay exceeds certain size

In this section, we proposed the VQ algorithm in order to support the DIVE applications. We conducted the experimental performance evaluations on the Differentiated Services network, where a DIVE application, MPEG-2 video and background traffic co-existed. Our analysis made clear that: 1) the existing DiffServ architecture will have difficulty to support DIVE applications; 2) the VQ based architecture fares very well, and is fully capable of supporting DIVE applications, while at the same time it protects the performance of other real time applications; 3) VQ is superior to an earlier proposed solutions; EF2. Our findings provide network related solutions, capable of supporting large-scale deployment of DIVE applications.

5.4 A Critical Assessment of The Presented Results

Using an experimental approach, we investigated the QoS support of MPEG-2 video on demand and DIVE applications over best effort and differentiated services architecture. Our work produced the following valuable contributions:

a). Engineering Value.

1). We proved that Linux-based boxes are able to provide QoS capable network nodes for small enterprises and the house environment.

2). We demonstrated the capability and limitation of DiffServ to support two of the most sensitive real time applications, MPEG2 and DIVE. Our research discovered that the architecture of DiffServ need to be modified for wide deployment of DIVE applications.

3). We provided clear quantitative relation between performance at the network level and application level. By observing packet loss at the in network level, you can understand the frame loss and other related parameters occurring at the application layer.

b). Hidden mechanism

1). We discovered the behaviour of DIVE applications in DiffServ networks and how it co-exists with other applications.

2). We discovered the weakness and difference between PQ and CBQ.

5.5 Summary

In this Chapter, we conducted a series of measurements to prove and verify the effectiveness of the DiffServ mechanism with the network layer parameters (the packet loss rate and the packet forwarding delay) and the application layer parameters (PSNR, the frame inter-arrival time, buffer occupancy etc).

6. Conclusions and Suggestion for Future Research

The explosive growth of communication technologies and deployment of networking infrastructure around the globe brought the Information Technology (IT) age. Internet based applications are becoming ever present in all our activities, such as Internet telephony, videoconference and E-commerce etc. There is no doubt that the Internet will play an ever-increasing role in our lives.

However, the increasing traffic volumes create congestion, deteriorating the quality of serviced applications. The current IP (Internet Protocol) network is a packet switched data network that provides best effort (no discrimination) service to all streams. Therefore, it is inevitable that packet of loss and delay sensitive applications experience

these impairment due to network congestion. This behavior does not affect considerably textual applications such as FTP, WWW etc, but it is intolerable for real-time applications such as Internet telephony and videoconference because of packet loss and delay. Even if we increase the bandwidth to Gigabits, the traditional best effort network still cannot guarantee to the real-time traffic certain packet loss rate and that packet will be arriving at the destination in the proper order and within the required time.

In order to enable Internet to support widespread servicing of real time applications, it is important that QoS technologies is proposed and deployed.

The DiffServ architecture can provide a QoS platform for real-time applications. However, there is no detailed information about the performance of real-time applications in DiffServ enabled IP networks, thus through studies are required.

This thesis is concerned with the service of real-time applications such as MPEG video and virtual reality (DIVE) applications over the highly scalable Differentiated Services networks environments. We tested MPEG-2 based video on demand under a variety of network conditions and examined the best effort and DiffServ architecture. At the same time, we examined the performance under different types of traffic and traffic volumes, using a variety of network architectures. We also linked the application level QoS parameters (frame loss and frame jitter) to network layer parameters (packet loss and packet jitter). This could allow a direct interpretation of the application's quality by simply observing the behavior of the packets at the network layer. Two main packet-forwarding disciplines: Priority Queuing (PQ) and Class Based Queuing (CBQ) were examined in our testing.

Through the PSNR calculation and video packet loss and delay analysis, we demonstrated that the DiffServ architecture could provide satisfactory QoS guarantees to an MPEG2 video streaming application. In addition, the experimental results showed that the existing DiffServ architecture cannot support effectively DIVE applications. To resolve this weakness, we proposed a new packet forwarding mechanism, named as

"Virtual Queue" (VQ). The VQ algorithm provides additional protection to the DIVE running under DiffServ. The algorithm makes servicing these loss and delay sensitive applications through Ethernet possible. Taking into consideration the importance of DIVE applications to electronic commerce, telemedicine, tele-learning etc., it is quite evident that the deployment of the extended architecture in Internet will be highly beneficial.

It is evident that the DiffServ architecture builds a solid QoS platform for various real-time applications. The test results also indicate that CBQ is more beneficial to real-time applications (for premium service) than PQ.

In addition to the scientific and engineering contributions made, our work proved that we are capable to provide inexpensive advanced networking equipment using off-the-shelf computing devices and open source software. These networking solutions can be very useful and can be used for developing a networking environment able to support even highly sensitive real time applications such as multimedia and virtual reality.

Future Research Work

Although we made a considerable effort, there are still many remaining QoS issues related to the DiffServ architecture. Nonetheless, our work has provided a solid and convincing platform for our future QoS research work.

Our work involved the use of a single DiffServ domain and focused on the core router. Experimental work of similar nature could be performed over multiple domain DiffServ architecture in which case the behavior of gateway nodes (egress of one domain/ingress of next domain) and the "wrapping" policies becomes crucial. The gateways could become point of bottlenecks, thus degrading the performance of the applications, rendering use of DiffServ at within the domains meaningless.

For our work, MPEG-2 video on demand applications were based on the use of a single video stream. However, within core routers in Ethernet, multiple streams are expected to present. A study where multiple MPEG-2 video streams are used might be proven beneficial.

The Multi-protocol Label Switching (MPLS) architecture has been proven highly beneficial in terms of QoS support and traffic engineering in Internet. MPLS and DiffServ have merged, providing a combined architecture [RFC 3270]. Study of the applications under such architectures is beneficial.

Reference

[ADE02] Al-Hezmi Adel, "Differentiated Services on Linux ", TKN Internet-component WS 2001/02
URL: http:// user.cs.tu-berlin.de/~adelhazm/study/diffserv.pdf

[ANS96] ANSI, "Digital Transport of One-Way Video Signals-Parameters of Objective Performance Assessment", ANSI T1.801.03-1996, February 5, 1996.

[APO91] T. Apostolopoulos, "*Model for the Analysis of a CSMA/CD Channel for Data and Voice Applications*", *Computer* Communications, Vol. 14, No.2, March 1991, pp. 71-79.

[AWD99] D. Awduche, "*MPLS and traffic engineering inn IP networks*", IEEE Communications Magazine, December 1999.

[BLA98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, " *An Architecture for Differentiated Services*", Internet RFC 2475, 1998.

[BRA94] R. Braden, D. Clark, S. Shenker, "*Integrated Services in the Internet Architecture: an Overview*", Internet RFC 1633, 1994.

[BRA97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "*Resource Reservation Protocol (RSVP) – Version 1 Functional Specification*", Internet RFC 2205, 1997.

[BEN95] S. Benford, A. Bullock, C. Greenhalgh, R. Ingram and D. Snowdon, *"Collaborative Virtual Environments: User Interaction and Populated Information Terrain"*, in Proceedings of Imagina'95, Monte Carlo, February 1995.

[BUC00] S. Bucheli, J. R. Moorman, J.W. Lockwood, and Sung-Mo Kang. "Compensation modeling for QoS support on a wireless network", Technical report, Coordinated Science Laboratory, University of Illinois, Nov, 2000

[CAD93] "*Communication Architecture for Distributed Interactive Simulation*", (CADIS), Institute for Simulation and Training, Orlando, Florida, 28 June 1993.

[CADIS] Communication Architecture for Distributed Interactive Simulation (CADIS), Institute for Simulation and Training, Orlando, Florida, 28 June 1993.

[CSA97] Csaba Szabo, "*An Ethernet Compatible Protocol to Support Real-Time Traffic and Multimedia Applications*", Computer networks and ISDN Systems 29 (1997), pp. 335- 342

[CLA92] D. Clark, S. Shenker, L. Zhang, "*Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*", Proc. SIGCOMM'92, pp.14-26, Aug. 1992.

[DAL94] I. Dalgiç, W. Chien and G.A. Tobagi, "*Evaluation of 10Base-T and 100Base-T Ethernets Carrying Video Audio and Data Traffic*" Proc. of IEEE INFOCOM'94, Toronto, pp. 1094-1102.

[DARSE] Darwin Server, http://developer.apple.com/darwin/

[DEM90] A. Demers, S. Keshav, S. Shenker, *"Analysis and Simulation of a Fair Queuing Algorithm"*, Internet Working: Research and Experience, Vol. 1, pp. 3-26, 1990.

[DIFF0] Differentiated Services. http://www.ietf.org/html.charters/diffserv-charter.html.

[DIVE0] DIVE. http://www.sisc.se/dive/dive.html

[DUN89] J. Dunlop, "*Techniques for the Integration of Packet Voice and Data on IEEE 802.3 LANs*", Computer Communications, Vol.12, No.5, October 1989, pp.273-280.

[EBE99] J. P. Ebert, A. Willig, "A Gilbert-Elliot Bit Error Model and the Efficient Use in Packet Level Simulation", Technical report TKN-99-002, Telecommunication Networks Group, Technical University Berlin

[EBR02] Touradj Ebrahimi, Fernando Pereira: *The MPEG-4 Book.* Prentice Hall, 2002.

[EDW94] Francis Edwards, Mark Sculz, "*Performance of VBR Packet Video Communications on an Ethernet LAN: A Trace-Driven Simulation Study*", 1994 IEEE 13th Annual International Phoenix Conference on Computers and Communications, pp. 427- 433

[FEA02] N. Feamster and H. Balakrishnan, "Packet Loss Recovery for Streaming Video", *12th International Packet Video Workshop*, Pittsburgh, PA, April 2002

[FEN99] W. Feng, J. Rexford, "Performance Evaluation of Smoothing Algorithms for Transmission of Prerecorded Variable-Bit-Rate Video", IEEE Transaction On Multimedia, Vol. 1, No. 3, September 1999.

[FL951] S. Floyd, V. Jacobson, "*Link-sharing and Resource Management Models for Packet Networks*", IEEE/ACM Transactions on Networking, Vol.3 No. 4, August 1995.

[FL952] Sally Floyd. Notes on CBQ and guaranteed service, 1995

[FLO93] S. Floyd and V. Jacobson, " *Random Early Detection Gateways for Congestion Avoidance*", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp397 – 413, August 1993.

[FLO98] S. Floyd, M. F. Speer, "*Experimental Results for Class-Based Queueing*", Nov. 1998.

[GAL01] J. R. Gallardo, D. Makrakis, *"Dynamic Predictive Weighted Fair Queuing for Differentiated Services"*, proceedings of IEEE International Conference in Telecommunications (ICC 2001), Helsinki, Finland, June 2001

[GAL99] J. R. Gallardo, M. Hafid, L. Qin, D. Makrakis and A. Hafid, "*DIVE-Based Applications over IPv6-Capable Networks: Teletraffic Studies, Performance Analysis and QoS*" in proceedings of the 7th International Conference on Advances in Communications and Control (COMCON7), Athens, Greece, July 1999.

[GAL91] D. L. Gall, "MPEG: a video compression standard for multimedia applications," *Comm of the ACM.*, vol. 34, 1991, pp. 654-665.

[GAR02] G. Carrozzo, V. Chionsini, Stefano Giordano, Saverio Niccolini: "*QoS Evaluation of Real-Time Applications over a Multi-domain DiffServ Experimental Test-Bed.*" NETWORKING 2002: 1093-1098

[GUP94] S. Gupta and C. L. Williamson, "*An experimental Study of Video Traffic on an Ethernet Local Area Network*", Proc. IEEE Globecom'94, pp. 558-562.

[HAM97] M.Hamdi. "Lcfs queuing for real-time audio-visual services in packet networks." In Proceedings of 8th Int. Workshop on Packet Video (AVSPN'97), Aberdeen, UK, September 1997.

[HAS97] B. G. Haskel, A. Puri, A. N. Netravali, "Digital Video: An Introduction to MPEG-2", Capman & Hall, 1997.

[HEI99] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski, "*Assured Forwarding PHB Group*", IETF RFC 2597, June 1999.

[HOR00] E. Horlait, and N. Rouhana. "*Differentiated Services, and Integrated Services use of MPLS*" http://citeseer.nj.nec.com/update/327451

[IEE97] IEEE Computer Society. IEEE std. 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, Jun. 1997

[IEE99] IEEE Computer Society. Supplement to IEEE std. 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Higher-speed physical layer extensions in the 2.4 GHz band, Sep. 1999

[IETF0] IETF, http://www.ietf.org/

[ISO95] ISO/IEC 13818-2: 1995(E) MPEG-2 Specification

[ITU02] ITU-R Recommendation BT.500-11: "Methodology for the subjective assessment of the quality of television pictures." International Telecommunication Union, Geneva, Switzerland, 2002

[ITU26] ITU-T Recommendation H.263 (1995): "*Video Coding for Low Bit Rate Communication*".

[ITU32] ITU G.723 Speech Compression Technology for the ITU H.324 and ITU H.323 Videoconferencing/Telephony Standards

[JAC99] V. Jacobson, K. Nichols and K. Poduri, "*An Expedited Forwarding PHB*", IETF RFC 2598, June 1999.

[KAR99] A. Karve, "Quality of Service Options for Real Time Traffic", Network Magazine, Feb 1, 1999.
URL: http://www.networkmagazine.com/article/NMG20000509S0026/1

[KUZ98] A.Kuznetsov, Iproute2+tc package. Technical report, May 1998.

[LAI00] J. Laine, S. Saaristo, J. Lemponen and J. Harju: *Implementation and Measurements of Simple Integrated Media Access (SIMA) Network Nodes*. ICC 2000, New Orleans, USA, June 18-22, 2000.

[LAI01] J. Laine, J. Harju, J. Karjalainen, J. Lemponen and S. Saaristo: *Real-Time Traffic Measurements in a Differentiated Services Network*. ICC 2001, Helsinki, Finland, June 11-15, 2001.

[LIN00] P. Lin, B. Bensaou, Q.L.Ding, and K.C. Chua. "A Wireless fair scheduling algorithm for error-prone wireless channels", *Proceedings of ACM WoWMOM 2000*, 2000

[LEI94] S. M. Lei, T. C. Chen, and M. T. Sun, "*Video Bridging Based on H.261 Standard*," *IEEE Trans. on Circuit and System. For Video Tech.*, vol. 4, no. 4, Aug. 1994, pp. 425-437.

[LUS97] S. Lu, V. Bharghavan, and R. Srikant. "Fair scheduling in wireless packet networks", Proceedings of *ACM SIGCOMM 1997*, 1997

[LUS99] S..Lu, T. Nandagopal, and V. Bharghavan. "Design and analysis of an algorithm for fair service in error-prone wireless channels", *Wireless Networks Journal*, Feb. 1999

[MAN97] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang, "*Resource ReSerVation Protocol (RSVP) Version 1 - Applicability Statement - Some Guidelines on Deployment*", Internet RFC 2208, 1997.

[MPGIP] MPEG4IP: Open Source, Open Standards, Open Streaming http://mpeg4ip.sourceforge.net/

[NIC98] K. Nichols, S. Blake, F. Baker, D. Black, "*Definition of the Differentiated Services Field (DS Field) in the Ipv4 and Ipv6 Headers*", Internet RFC 2474, 1998

[QIN99] L. Qin, "*Quality of Service Management in Distributed Interactive Virtual Environment*", M.Sc. Dissertation, Computer Science, University of Western Ontario, 1999.

[OVE00] Overview of the MPEG-4 standard http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm

[RAD99] S. Radhakrishnan. Linux – advanced networking overview – version 1. Technical report, Information and Telecommunications Technology Center, Department of Electrical Engineering and Computer Science, The University of Kansas, Lawrence, KS 66045-2228, AUG 1999

[RAO90] K.R. Rao and P. Yip "*Discrete Cosine Transform---Algorithms, Advantages, Applications*" (Academic Press, London, 1990), ISBN 0-12-580203-X.

[RED00] RED parameters implementation on Linux http://www.ittc.ukans.edu/~pramodh/courses/linux_qos/qos_qcf.html

[RFC3270] - Le Faucheur F. et al, *MPLS Support of Diff-Serv* - RFC 3270

[RFC36] RFC 2836, http://www.faqs.org/rfcs/rfc2836.html

[ROS99] E. C. Rosen, A. Viswanathan, R. Callon, " *Multi-Protocol Label Switching Architecture*", Internet Draft, 1999.

[SCH96] H. Schulzrinne, S. Casner, R. frederick and V. Jacobson, "*RTP: A Transport Protocol for Real-Time Applications*", RFC-1889, 1996.

[STA99] W. Stallings, "*Data and Computer Communications*", 6th edition, ISBN 0-13-086388-2, Prentice Hall Co., 1999.

[STA02] G. Stattenberger and M. Scheidegger and T. Braun and Marcus Brunner and Heinrich Stüttgen: "*Performance Evaluation of a Linux DiffServ Implementation*" Computer Communications Journal, Elsevier, 2002

[STE94] Stevens, W. Richard "*TCP/IP illustrated: the protocols*", Addison Wesley Publishing Company, 1994. ISBN 0-201-63346-9 (v.1).

[SEC01] Chuck Semeria, "Supporting Differentiated Service Classes: Queue Scheduling Disciplines". Juniper Networks. December 2001

[TOB96] F. Tobagi, I. Dalgic, "*Performance Evaluation of 10BASE-T and 100BASE-T Ethernets Carrying Multimedia Traffic*", IEEE Journal on Selected Areas in Communications, Vol. 14, No.7, September 1996, pp. 1436- 1454.

[VID00] Video Quality Research Homepage: http://www.its.bldrdoc.gov/n3/video/Default.htm

[WAN95] Q. Wang, M. Green and C. Shaw, *"EM - an environment manager for building networked virtual environments"*, in Proceedings of the Virtual Reality Annual International Symposium. VRAIS'95, pages 11-18. IEEE Computer, IEEE Functional Specification, Draft prETS300652, ETSI Secretariat, July 1995.

[WER00] A. Werner, S. Jamal, K. Alexey. "Differentiated Services on Linux", June 1999. http://diffserv.sourceforge.net

[WER01] A. Werner, "Linux Traffic Control – Implementation Overview", Technical report, EPFL, Nov 1998. http://tcng.sourceforge.net

[WIS01] L. Wischhof, J. W. Lockwood, "Packet Scheduling for Link-Sharing and Quality of Service Support in Wireless Local Area Networks", Technical Report, WUCS 01-35, Washington University

[WUD00] D. Wu, T. Hou, B. Li, W. Zhu, Y.-Q. Zhang, H. J. Chao, "An End-to-End Approach for Optimal Mode Selection in Internet Video Communication: Theory and Application," *IEEE Journal on Selected Areas in Communications* (JSAC), Special Issue on Error-Resilient Image and Video Transmission, vol. 18, no. 6, pp. 977-995, June 2000.

[XIA99] X. Xiao and L. NI, "Internet QoS: A Big Picture", IEEE Network Magazine, March/April, pp. 8-18, 1999

[XIA00] X. Xiao, A. Hannan, B. Bailey, "*Traffic Engineering with MPLS in the Internet*", IEEE Network, March/April 2000.

[YAT01] T. Yang, Z. Chen, D. Makrakis, A. Hafid, "*Study of Assured Forwarding and Expedited Forwarding Service*", proceedings of the IEEE 15th International Conference on Information Networking (ICOIN-15), Beppu, Japan, Deb. 2001.

[YU011] H. Yu, Q. Zhou, D. Makrakis, N. D. Georganas, and E. Petriu, "*Quality of Service Support of Distributed Interactive Virtual Environment Applications in IP Networks*", Proceedings of Pacific Rim 2001 Conference, Victoria, Aug. 2001.

[YU012] H. Yu, D. Makrakis, L. Orozco-Barbosa, *"Experimental Evaluation of MPEG-2 Video over Differentiated Services IP Networks"*, proceedings of IEEE Pacific Rim Conference 2001, Victoria, BC, Aug. 2001.

[YUH01] H. Yu, D. Makrakis, L. Orozco-Barbosa, *"Experimental Evaluation of MPEG-2 Video over Differentiated Services IP Networks"*, proceedings of IEEE Pacific Rim Conference 2001, Victoria, BC, Aug. 2001.

[ZDJ89] J. Zdepski, K. Joseph, D. Raychaudhuri, "Packet Transport of VBR Interframe DCT compressed Digital Video on a CSMA/CD LAN", IEEE Globecom'89, Vol.2, pp. 886- 892.

[ZH021] Quanyou Zhou, Hong Yu, Dimitrios Makrakis, Ognian Kabranov *"Performance Evaluation for Real Time Traffic Over Differentiated Services Capable IP Networks"*, CCECE 2002, Winnipeg, May.2002

[ZH022] Quanyou Zhou, Hong Yu, Dimitrios Makrakis, Nicolas D. Georganas, Emil Petriu. "*Performance Evaluation for Distributed Interactive Virtual Environment Applications over IP-DiffServ Networks*", IEEE International Workshop on Haptic Virtual Environments and their Applications, Ottawa, Canada, November, 2002