Optimal Link Utilization and Enhanced Quality of Service Using Dynamic Bandwidth Reservation for Pre-recorded Video

Mukul Kishore

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters of Science
in
Electrical Engineering

Dr. Yao Liang, Committee Chair
Dr. Luiz DaSilva, Committee Member
Dr. Saifur Rahman, Committee Member

Date: October 31st, 2003
Alexandria, Virginia

Keywords: MPEG, QoS, RSVP, RCBR, Video on Demand, SBM

Optimal Link Utilization and Enhanced Quality of Service Using Dynamic Bandwidth Reservation for Pre-recorded Video

Mukul Kishore

(Abstract)

Video-on-Demand (VoD) is a service that allows people to request and view stored videos or movies of their choice directly online from a VoD service provider. The selected streaming videos are then delivered over the broadband Internet. The bursty nature of Variable-Bit-Rate (VBR) compressed video (such as MPEG) poses some important issues for video delivery over high-speed networks due to its significant bit rate variation over multiple time scales. However, sufficient quality of service (QoS) mechanisms must be in place before it can be widely enabled and deployed over Internet.

Conventionally a static bandwidth level close to the peak rate is reserved for a streaming video flow. Any static allocation of network resources for VBR video traffic would be difficult and inefficient considering the peak rate to be significantly higher than the average data rate. Since the traffic pattern over time is already known for pre-recorded videos, this issue is addressed by the Renegotiated Constant Bit Rate (RCBR) service which proposes QoS allocation over multiple time scales. Since this mechanism has been tested via simulations and analysis only we implemented it on a real test bed with a VoD server and clients to study its performance. We observed that under heavy bandwidth constraints the performance of RCBR is much better than traditional CBR in terms of packet loss rate. We also implement a new Adaptive Buffer Window mechanism and the concept of application level smoothing to increase the scalability of a VoD server.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

CHAPTER 1 INTRODUCTION

Video on Demand (VOD) services are amongst the list of future services with significant growth potential. They interest customers because more and more households all over the wired world are finding it more affordable than ever to install a broadband Internet connection. The bandwidth rates have fallen considerably during the past three years due to investments by the Telecommunication industry during the dotcom boom days. Most hardware vendors are now making the average home PC into a multimedia center to play music, videos and gaming. All this has fueled the growth of most entertainment sectors such as radio stations, news channels, television broadcasts into the digital realm and encouraged altogether phasing out the need for more electronic apparatus except the common home PC.

Streaming media is well known to be computation and network resource hungry. A lot of research has been done to optimize the digital encoding, storage, data transfer and playback aspects of the same. In certain realms however scope still exists for further improvement and optimization. This thesis work attempts to enhance the network bandwidth reservation aspect and proposes some useful bandwidth adaptation features.

1.1 Overview and Motivation

Streaming video relies on both high-speed networking and data compression technologies to be used satisfactorily. Data compression introduces burstiness in the video stream that complicates the server and network resource utilization. Another important constraint for streaming video is that quite often decisions need to be made online so that the video stream may adapt to the best performance without unwanted processing delays. Since streaming media is very delay sensitive a lot of research work has been done towards mitigating the problems faced by video feeds. Some of the research work worth mentioning includes statistical multiplexing [CH93, RRB93], smoothing with delay tradeoff [LCY94], jitter control [SJ93, ZFV92] and adjusting the quality of service to fit the resources available [PE92]. Pre-recorded video applications can take a more flexible

approach to latency of data delivery because the entire video stream is known a priori. It is possible to calculate a complete plan for delivery of video data that avoids both the loss of picture quality and waste of network bandwidth due to over provisioning of network resources [FJS97].

1.2 Research Objectives

Most of the above mentioned research work is based on the MPEG-1 standard and dates back to late 1990's. Subsequent to that the fast evolving research areas included digitization and encoding standards (such as MPEG-4 and MPEG-7) and optimization and prediction of real time streaming video (such as live broadcasts). Most of prerecorded Video-on-Demand (VoD) service providers still use digitized MPEG-1 or MPEG-2 encoded videos that they stream over dedicated cable networks not involving the Internet Protocol (IP). Some research has been done for streaming video technology over shared IP networks [MSD97, CK96] that aims at optimizing the bandwidth reservations in terms of Quality of Service (QoS) guarantees. However almost all the research involving QoS for VoD services took advantage of simulation and theoretical models and not real test beds.

These are my principal research objectives:

1. To implement a test bed containing servers, clients and intermediate routers working with the Resource Reservation Protocol (RSVP) and the Subnet Bandwidth (SBM) protocol.

2. Use the test bed for testing an improved bandwidth negotiation scheme called RCBR (Renegotiated Constant Bit Rate) [MSD97] that dynamically updates bandwidth reservation level as a video advances in real time. This is done by reading the traffic trace files of prerecorded videos.

3. Through theoretical analysis and simulation, determine the bandwidth savings incurred by using dynamic bandwidth reservation.

4. Observe and compare the packet loss rate on a network test bed for static and dynamic reservation implementations.

5. Propose a new agent as an add-on to the functionality provided by the Designated Subnet Bandwidth Manager (DBSM) which is a network agent responsible for admission control of QoS requests.

6. Propose a new Adaptive Buffer Window (ABW) mechanism to dynamically search for the best sending buffer without overloading the server packet shaper module.

1.3 Organization of the Thesis

The next chapter provides most of the background information needed to understand various technologies referred to and used in this research, namely MPEG encoding standards, Video on Demand technology and Quality of Service mechanisms. Chapter 3 provides theoretical analysis and discussion of advantages incurred by using a dynamic bandwidth reservation for MPEG video streams. It also carries preliminary analysis results and plots of different video types (with calm or intense screen activity). Chapter 4 carries a description of the test bed setup and software code used to gather results. Chapter 5 contains results, plots and analysis. Chapter 6 contains conclusions and future areas of research.

CHAPTER 2 Background

The Video-on-Demand service allows people to request and view stored videos or movies of their choice directly online from a VOD service provider. The selected streaming videos are then delivered to viewers over the broadband Internet. Video on Demand technology had existed for almost a decade when the first patent for broadcasting movies on demand was acquired by USA Video [www.usvo.com]. Conventionally, Video on Demand services have mostly existed within the cable TV domain with set-top boxes managing the requests from a menu provided by the cable operator. After the simple request process, the video data is sent over the designated cable TV channel using the standard video feed. This model has been changing with the advent of new companies such as CinemaNow® [www.Cinemanow.com], Intertainer® [www.Intertainer.com] and MovieLink® [www.movielink.com], which have started providing customized video feeds through the public Internet. They send the digitized video using the Internet Protocol (IP) as a bit stream out of the encoded MPEG version of the movie file. After purchasing the video feed online the user must wait till the movie has been downloaded in full or substantially buffered to start a stream.

The video stream mostly comes out of the local user's buffer after the initial download time because of the delay restrictions that exist for watching motion video online. This is because if video streams are watched streaming, they face the issues of jitter and lost frames. These issues exist assuming the video stream travels over a publicly available network such as the Internet or a shared network of a private service provider. To provide better than best-effort service to customers who are ready to pay more for a guaranteed level of service, various Quality of Service (QoS) mechanisms are available to reserve a certain chunk of bandwidth for one particular video stream. Well known amongst these QoS mechanisms are MPLS (Multi Protocol Label Switching) and RSVP (Resource Reservation Protocol). These mechanisms enable prioritization of particular (perhaps more paid-for) data flows over other (perhaps less paid for) flows over a common data channel. The resources that have to be reserved on a network must be decided by looking at the expected traffic requirements of a data stream.

Being a Variable-Bit-Rate (VBR) compressed video technology, a MPEG video stream can exhibit significant bit rate variations over multiple time scales. Usually a peak traffic rate characterizes the reservation level for the entire duration of the file transfer. This peak rate is obtained by looking at the traffic profile of the encoded video and the chosen rate corresponds to the burstiest part of movie transmission. In most cases this is set as the reserved rate of video transfer. Sending this bursty VBR video at the constant peak reservation level poses some important issues for video delivery over high-speed networks. On the one hand, we want to provide satisfactory QoS for video traffic, for example, allocating network bandwidth close to the peak rate of the entire transmitted video; and on the other hand, we want to utilize network resources efficiently, due to the fact that for any bursty video source, its peak rate would be significantly larger than its average rate. Intuitively the bandwidth allocation at peak rate would be wasteful for resources.

The concept of RCBR (Renegotiated Constant Bit Rate) [MSD97] stream proves to be quite efficient in removing the wastefulness due to a constant bandwidth reservation level. The concepts necessary to understand this thesis work are:

- MPEG technology
- Video on Demand technology
- Network Quality of Service mechanisms such as IntServ (Integrated Services) and DSBM (Designated Subnet Bandwidth Manager).

These are described in this chapter.

2.1 MPEG Encoding Technology

MPEG (pronounced M-peg) is an acronym for Moving Picture Experts Group and it is the name of the family of standards used for coding audio-visual information (e.g., movies, video and music) in a digitally compressed format. This family of standards includes MPEG-1, MPEG-2, MPEG-3 and MPEG-4 standards. The MPEG-1 and MPEG-2 standards are mostly used for video content and MPEG-3, more popularly known as MP3, is used for audio formats. MPEG-4 is the new generation of encoding algorithms that can be used specifically for multimedia applications. The Moving Pictures Experts Group is listed under the ISO/IEC working groups to develop international standards for compression, decompression, processing and coded representation of motion pictures, audio and a combination of both pictures and audio [MPGURL].

As of present, the MPEG working group has produced the following standards:

MPEG-1: A standard for storage and retrieval of moving pictures and associated audio on storage media

MPEG-2: A standard for digital televison

MPEG-3: A standard for encoding music files

MPEG-4: A standard for multimedia applications

MPEG-7: A content representation standard for information search

Within the MPEG working group the MPEG video group is the largest working group. It is authorized to develop and standardize video coding algorithms and tools. The MPEG video group successfully completed the MPEG-1 video coding standards in 1992 and MPEG-2 video-coding standard in 1994. It finalized the MPEG-4 video standard in 1998.

The MPEG-1 video compression standard supported data rates of digitally stored media at 1 to 1.5 Mbps. This standard is well suited for a range of applications at a wide variety of bit rates. It also allows real time decoding of a bit stream and also supports features that facilitate interaction with the stored bit stream. Along with developing a standard for digital storage media for use by technologies such as CD-ROMs, MPEG-1 was developed as a generic standard that can also be used in other digital video applications such as telecommunications. The MPEG standard definition has three parts:

- Synchronization and multiplexing of audio and video
- Video – representation and coding
- Audio – representation and coding

The video representation and coding part is the one relevant to the research presented in this thesis. This is because the video coding and the video frame structure (whose description follows) are the parameters taken into consideration while renegotiating bandwidth. Uncompressed digital video is a sequence of still images and thus it requires an extremely high transmission bandwidth. A good example would be that of a digital video represented using the NTSC (North America Television Standards Committee) resolution, which has a bit rate close to a hundred megabits per second. For multimedia applications it becomes absolutely essential to reduce the bit rate by using compression techniques to make the video usable at all for display or transfer. The compression technique uses the spatial and temporal redundancy that is present in any digital video signal. Compression algorithms make the process inherently lossy by introducing artifacts into the decoded signal. This causes the reconstructed signal from the compressed bit stream to differ from the input video signal. The MPEG video standard requires the highest possible quality of decoded video at any given bit rate. In addition to the high quality of the decoder output, the applications using the video necessitate some resilience to bit errors and sufficient support to varying video formats.

2.1.1 MPEG-1 Video Compression algorithm

The MPEG-1 video compression algorithm uses the JPEG (Joint Pictographic Expert Group) still image compression standard (ISO 1991) combined with the H.261 video conferencing standard (CCITT 1990). Since a sequence of still images creates a video signal, it is possible to use the JPEG compression technique to encode a video signal. This compression method where each frame of video is individually and encoded into an independent entity is referred to as ***intraframe*** coding.

This intraframe coding reduces the image size by considering the spatial redundancy that exists between the adjacent pixels of a still picture frame. Inspired from the JPEG and H.261 video encoding model, the MPEG video-coding algorithm utilizes a two-dimensional block-based Discreet Cosine Transform (DCT). In this technique a picture frame is first divided into a grid where pixel blocks of size 8x8 are chosen and a two dimensional Discreet Cosine Transform is applied independently to each block. This generates an 8x8 block of DCT coefficients where most of the energy in the original block is concentrated in a few low-frequency coefficients. Many of the DCT coefficients are set to zero when a quantizer is applied to each of them. This quantization causes the compression algorithm to become lossy and only the non-zero coefficients after the quantization operation are transmitted. Further compression is achieved by entropy (Huffman) coding of runs and amplitudes of these coefficients.

The JPEG encoding technique is responsible for the ‘intraframe’ coding and this alone does not suffice for producing a bit rate of 1.5Mbps, so the H.261 algorithm is used as an ‘interframe’ coding technique. It is used to exploit the high degree of inter frame redundancy and correlation existing between adjacent frames of an MPEG encoded video. The H.261 algorithm computes a frame-to-frame difference signal, called the prediction error, in calculating which technique of motion compensation is employed to correct for the motion aspect of the video. This motion compensation technique is realized by using a block based approach where a target block, which is a block of pixels, in the frame to be encoded is chosen and this block is compared with a set of similar blocks of the same size in the previous frame, referred to as the reference frame. The

block in the reference frame that is the closest matching to the target block is chosen and used as prediction for the latter. The difference or displacement between the target frame and the reference frame is represented as the prediction error. The best matching block is associated with a motion vector that describes the prediction error between the reference block and the best matching block. The prediction error itself is encoded using the DCT-based intraframe encoding technique and the motion vector information is transmitted along with it after being encoded. In MPEG video, the H.261 algorithm is used and the block size for motion compensation is chosen to be 16x16 pixels. By increasing the block size the compression factor increases but it also comes at a tradeoff of the increase in cost associated with transmitting the motion vectors. The value of 16x16 represents a suitable level considering the tradeoff mentioned.

Another key feature of MPEG video compression is **bi-directional temporal prediction** also referred to as motion compensated interpolation. Using bi-directional temporal prediction, some of the video frames are encoded using two reference frames instead of just one. Out of these two reference frames, one is in the past and one is in the future. From the frame to be encoded, some target blocks are associated with a motion vector to a frame in the past called the past reference frame and this is called forward prediction. When the target frame is associated to a future reference frame, it is called backward prediction. Sometimes an average of two blocks, future and past, is used and this is called interpolation, where two motion vectors related to two reference frames are used. Bi-directionally predicted frames can never be used as reference frames and thus can never contain reference blocks.

A primary advantage of using bi-directional prediction is that compression obtained is higher than that obtained from forward prediction. These bi-directional encoded frames use less coding bits for the same picture quality than those just using forward prediction. There is a tradeoff of an extra delay in the encoding process since frames must be encoded out of sequence and this is contrary to the real time decoding policy of MPEG. Some extra encoding complexity is incurred since block matching must be performed twice for each target block. After receiving the frame to be decoded, the decoder scans

ahead and deciphers the correlation existing between the to-be decoded frame and the forward prediction and reverse prediction frames. It then maps the motion vectors contained in the 16x16 motion compensation block to predict the frame content in terms of 8x8 pixel blocks containing Discreet Cosine Transform coefficients and then finally decodes the characteristics of pixels inside the block.

2.1.2 MPEG bit stream

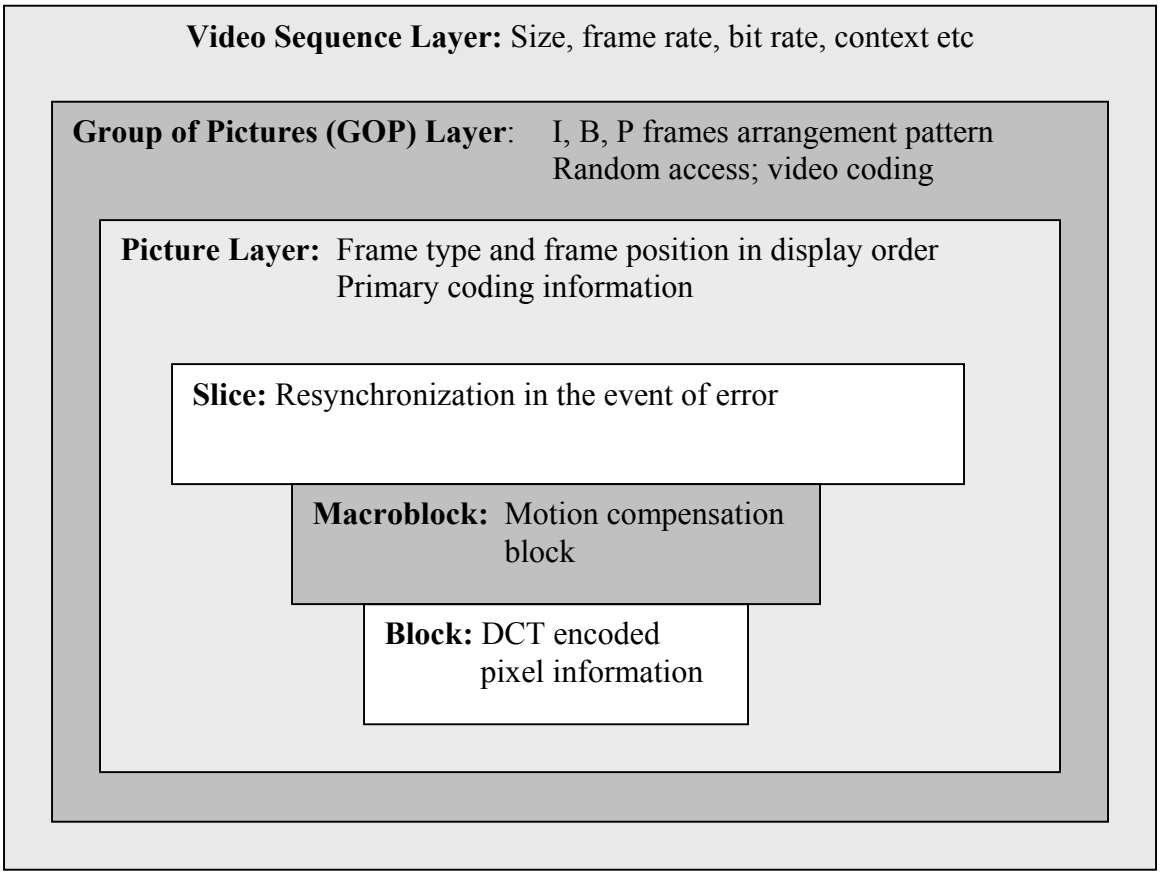


Figure 2.1 : A list of various layers used in the MPEG bit stream syntax

The MPEG bit stream is designed to be flexible enough to support a variety of applications. The syntax of the bit stream is arranged in several layers, each performing a different logical function. The layers are kept so that the bit information in the

macroblock motion vectors and the DCT encoded blocks is wrapped in the layers describing the Group of Pictures (described later) and picture layer.

The outermost layer is called the Video Sequence Layer and it contains the basic parameters for the MPEG encoded file such as the characteristics that define the size of the video frames, the frame rate, the bit rate and certain other global parameters. These parameters take a wide range of valid values. The video sequence layer encloses the Group of Pictures (GOP) layer, which provides support for random access, fast searching and editing of the movie file. A group of pictures provides an arrangement of frames in a deterministic order of the I (intra-coded frame), P (forward predictive coded frame) and B (bi-directionally predictive coded) frames. A sequence of frames is divided into a series of GOPs. A sample GOP pattern, which is the one used for my research, is presented diagrammatically below. The GOPs can be of any arbitrary structure and length and may be used to adapt to various data rates and compression ratios.

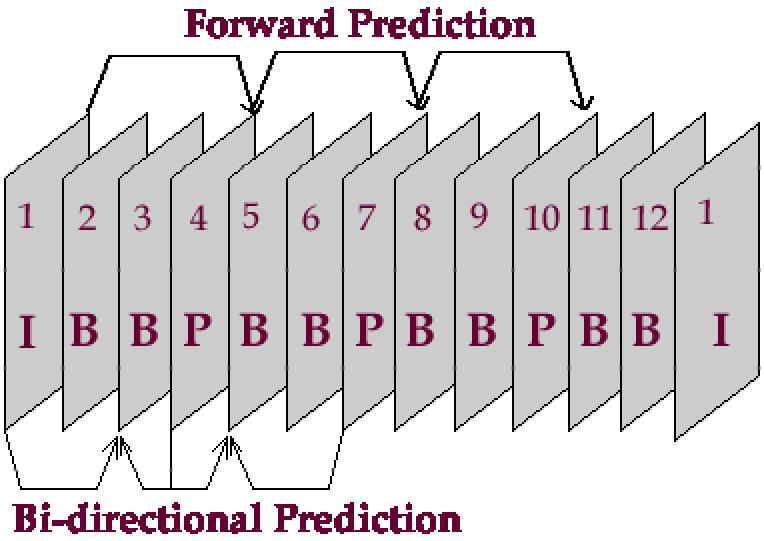


Figure 2.2: I, B and P frames in a Group of Pictures (GOP)

The bits produced after encoding a single frame of a GOP constitute the Picture Layer. This picture layer contains information about the type of frame present (Intracoded I, Forward Predictive coded P, Bidirectional Predicted frame B) and the position of the

frame in the order of display. The next set of layers inside the picture layer constitutes of the Slice layer, the Macroblock layer and the Block layer. The Block layer contains the 8x8 DCT unit and the Macroblock layer contains the 16x16 motion compensation unit that contains the motion vectors to other frames for interframe coding. The intention of using the Slice layer is for resynchronization during frame decoding in case there are any bit errors. Prediction registers used during differential encoding of motion vectors are reset at the start of a slice. The encoder is responsible for choosing the appropriate length of each slice. Inside the Macroblock layer, the block layer containing DCT coefficients of the 8x8 blocks in the Macroblock follows the motion vector bits for a Macroblock.

2.1.3 MPEG – 2

MPEG currently consists of two operating specifications, MPEG 1 and MPEG 2. MPEG-1 was developed for progressive source materials like film, while MPEG 2 was enhanced to address the interlaced materials common in broadcast TV. Both standards include video, audio, and systems components such as time stamping for synchronization. MPEG-1 defines a bit stream syntax for compressed audio and video optimized to not exceed a bandwidth of 1.5 Mbit/sec. The bandwidth restrictions fit the capabilities of single speed uncompressed CD ROM and Digital Audio Tape. Many people have taken the bandwidth design goal as a fundamental limit to MPEG 1 capability, but that is not the case. MPEG 1 defines the ability to process fields up to 4095 x 4095 and bit rates of 100 Mbit/sec. As a practical systems tradeoff, many suppliers produce systems capable of much lower levels of resolution. Although MPEG-2 follows the same generic MPEG encoding and predictive technology, it has some differences from MPEG-1. MPEG-2 can represent interlaced or progressive video sequences, whereas MPEG-1 only supports progressive. In MPEG-2 the aspect_ratio_information refers to the overall display aspect ration (e.g. 4:3, 16:9), whereas in MPEG-1 the ratio refers to the particular pixel. This reduction in the entries of the aspect ratio table helps interoperability by limiting the number of possible modes. In interlaced sequence, the coded macroblock height of a picture must be a multiple of 32 pixels while the width is a coded multiple of 16 pixels just like MPEG-1. The Group of Pictures (GOP) layer is totally optional for MPEG-2 and it mandates that all macroblock rows must start and end with at least one slice. In MPEG-

1, it was possible to code a picture with only a single slice. The mandatory slice per macroblock row restriction also facilitates error recovery.

2.2 Video on Demand Technology

2.2.1 Introduction

The broadband Internet is an ideal environment for distributing multimedia applications such as voice, video and interactive games amongst the users in an inexpensive way without installing any special network infrastructure for the same. With the constant growth of Internet subscribers and that of demand for distributed multimedia services, developing such applications on a best effort network such as the public Internet needs special consideration in terms of bandwidth, timing and buffering requirements of multimedia data [KT99].

Video-on-Demand (VoD) is a service that allows people to request and view stored videos or movies of their choice directly online from a VoD service provider. The selected streaming videos are then delivered to viewers over the broadband Internet. In this kind of service, a Video-on-Demand server provides video services where a user can request a specific kind of video of his/her choice at any time. This video information maybe a movie on demand, a remote learning course, an interactive news feed or a shopping video catalog. This multimedia network service holds great potential to challenge the multi-billion dollar video rental industry. However, sufficient QoS mechanisms must be in place before such Video-on-Demand service can be widely enabled and deployed over Internet. These mechanisms will be discussed in detail in the following sections.

During early years of VoD on the Internet, the video files were required to be downloaded and saved to the local client system before viewing could begin.Since most recently streaming audio and video have become available from live sources on the web, it has become customary that the viewer can start viewing the content within seconds of

transmission commencement and within the arrival of some first few bytes of media at the client machine.

2.2.2 Use of Streaming Technology

Primarily two popular methods of video streaming emerged as the demand for streaming content increased. The first method was the web server method, where a standard web server is used to supply data to the client using HTTP. This method is quite close to the download and play model. The compressed media file is made available as a link on the regular web server and via this URL the client's media player launches and plays the media clip from its buffer as the download progresses. Several streaming media formats such as Microsoft's ASF (Advanced Streaming Format) use this technology. Since there is a buffer involved, it implies some starting delay and continuous playback in case of congestion in the network (provided the congestion delay and data rate product does not exceed the buffer capacity).

TCP implements flow control mechanisms such as the 'slow start' mechanism, which keep increasing the sending data rate till network congestion is detected before bringing the rate back to a low initial value and restarting this process. The disadvantage of this approach for delay sensitive video streams is that TCP's aim is to send all the data reliably without considering packet latency. Packets delivered reliably but after their playback instant are of no use to the video player.

The other approach is the streaming media server approach where after the initial steps of request and reply, the server sends the data actively and intelligently to the clients. This quite differs from the passive approach used by TCP based protocols as the video server delivers the content at the exact data rate associated with the compressed audio or video stream. This is possible as the video stream could exhibit highly variable bit rate but constant frame rate. The number of bits contained in the number of frames may vary at all time levels. Control mechanisms may exist to maintain server and client synchronization during the data streaming process. Sending media data using UDP (User Datagram Protocol) is always a popular choice for multimedia traffic as UDP is a

lightweight protocol without any flow control functionality of its own. This makes UDP an ideal protocol for transmitting real time audio or video data. Some key features why UDP is preferred over TCP for sending video data can be listed as:

1. UDP traffic gets an advantage over TCP traffic on the Internet owing to the back-off policies that are implicit in the TCP protocol
2. UDP retransmissions can be controlled by the sending application so as to have any desired flow and quality control in accordance with the synchronization maintained amongst the video server and the clients.
3. Unreliability of the UDP protocol can enable lost packets to be counted as lost frames and thus adjust the screen resolution for the client. Streaming protocols such as RTP and RTCP may be used for appropriate sequencing
4. Lack of any flow control in UDP does not cause the latency and jitter problems resulting from use of TCP's slow start, time out and retransmission features.
5. A streaming UDP server can scale better than a HTTP based server
6. Multicasting is popularly implemented through UDP only though recently there have been some proposals of reliable multicasting using TCP.

Successful operation of any streaming VoD service requires highly scalable video servers that scale in both bandwidth management and storage capacity. It must have an efficient set of distributed modules such as:

1. Protocol specific streaming
2. Requested protocol adaptation
3. Media conversion or decompression
4. Media caching and formatting
5. Tape or disk storage
6. Negotiator modules for optimal media routing and scheduling
7. Negotiator modules for bandwidth reservation

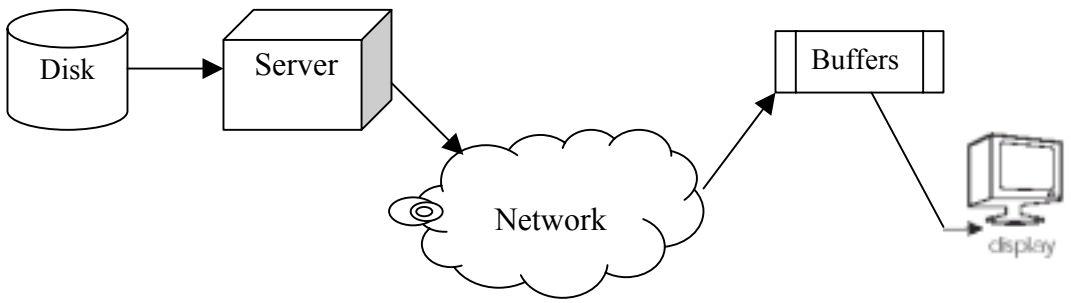


Figure 2.3: A typical distributed Video on Demand system

The distributed modules attain scalability and platform compatibility by adapting their outgoing data according to the supported protocol and memory availability. A powerful home PC with a broadband internet connection on a private VOD network may be fed with a higher resolution video MPEG-2 using TCP; a mobile handheld device, on the other hand, could be sent highly compressed MPEG-4 data with low resolution using RTP/UDP at a much lower bit rate [DS97].

Typically, the VOD server retrieves data from its disk subsystem into its memory buffers from where it is transmitted into the network according to some transmission schedule. This transmission schedule is computed in a manner so as not to starve the client buffer for data and neither overflow the same with yet-to-be-displayed data. Such transmission schedules can use, for example, the RCBR (Renegotiated Constant Bit Rate) [MSD97], PCBR (Piecewise Constant Bit Rate) [CK96] concept and Optimal Smoothing algorithms [FJS97]. RCBR and PCBR algorithms use the knowledge of MPEG traffic in advance to set the sending rate as not to overwhelm the network with data or starve the network for data. The optimal smoothing algorithm suggests the intelligent use of buffering to adjust the sending rate to a value close to the average rate and enhances utilization through statistical multiplexing of various simultaneous flows. The research presented in this thesis attempts to realize the RCBR concept by experimenting with real world traffic and renegotiation using RSVP. So far, mostly theoretical and simulation oriented work has been done to prove the efficiency of the RCBR concept and moreover most of it is related to data flow control and less to the bandwidth reservation level. In this thesis, the bandwidth reservation concept is realized by using a real client and server model that

works on the RCBR principle to change the reservation level dynamically as the video stream is fed to the client by looking at the traffic trace of the video being streamed.

2.2.3 Need for Bandwidth Reservation

Multimedia applications generate traffic at varying rates and require network throughput comparable to the rate of generation. Additionally, applications have a variable tolerance to latency and loss of data. These requirements may be summarized using the following parameters:

- **Throughput**: End to end data rate between sending and receiving entities
- **Latency**: End to end delay experienced by a packet sent over a network
- **Jitter**: Variation in latency
- **Loss**: Data packets dropped while in transit

If infinite network resources were available, all application traffic would be carried at the desired throughput bandwidth with no delay, zero jitter and zero packet loss; however, bottlenecks are often present in the network, which may be unable to satisfy one of more of the above listed parameters. The single most important issue against growth of online multimedia has been lack of reliable bandwidth to transport the data. Network congestion due to long router queues introduce lost frames and long latency for successfully delivered frames. This may cause problems of jitters leading to visual quality impairments such as picture freezes and low video resolution (in case the decoder decides to drop frames intelligently after detecting collision). Thus the use of a best-effort network for multimedia traffic results in unreliable behavior, which may be avoided by reserving bandwidth for that particular flow.

QoS (described in detail in the following section) is achieved primarily by two ways; (1) *packet-marking*, by classifying priority flows by setting special bits on a packet's IP header or, (2) *signaling* i.e., sending control packets to create (and update) reservation states for specific flows. Both these layer 3 reservation methods need QoS aware routers that can read and understand packet markings or signaling messages. The fact that routers

are not QoS enabled by default is a plausible argument against the widespread use of any QoS technology dependant upon router compatibility. However, it is safe to assume that at any particular privately owned network such as a corporate organization, a campus LAN or a neighborhood cable-Internet network, it is reasonably simple to manage and configure all the routers so as to support QoS mechanisms. Since any of the above mentioned networks carry different levels of prioritized traffic, it becomes worthwhile to separate priority traffic from general best-effort background traffic.

2.3 Quality of Service

2.3.1 Introduction

According to Yoram Bernet, author of the book *Networking Quality of Service and the Windows Operating System* [BER], 'Network QoS' is defined as –

> "The *capability to control traffic-handling mechanisms* in the network such that the network meets the service needs of certain applications and users *subject to network policies."*

The italicized words emphasize the fact that the goal of a QoS enabled network is not necessarily to optimize service to each application and user individually but rather its goal is to maximize the utility of the network across all applications and users. Network QoS provides mechanisms to control the allocation of resources among applications and users. Individual needs of an application or a user are an important consideration, as many applications require a minimum level of resources. If the minimum resources required cannot be provided to a certain application instance, it may be preferable to deny any resources to it at all and instead put resources to a better use by a less demanding application.

2.3.2 Traffic classifying mechanisms

Data traffic is handled over any shared network using certain protocols specific to the OSI layer handling the data. The underlying operation of any traffic handling mechanism is derived from queuing theory principles and algorithms such as FIFO, fair queuing,

weighted fair queuing, hierarchical fair queuing etcetera. Some examples of traffic handling mechanisms are described below.

802 user_priority

This refers to the traffic handling mechanism used for local area networks (LANs) based on the IEEE 802 technology. Most of the LAN technologies such as Ethernet, Token Ring, FDDI and other shared media networks are based on IEEE 802 standards. This IEEE standardized traffic handling technology defines a field in the Layer 2 header of 802 packets, which carries one of eight priority values. Sending entities such as hosts and routers typically mark each packet they send into the LAN with one of these priority values. Layer 2 devices such as bridges and switches are supposed to treat the packets appropriately using their underlying queuing mechanisms and Layer 3 devices are not affected with this marking at all. Since 802 user_priority is cheap to implement on the ubiquitous LANs, it is a promising technology however the latencies introduced by LAN switches may create problems for other traffic

Differentiated Services (DiffServ)

Since IP has emerged as the network layer protocol for both Internet and intranet, it is safe to focus only on IP-centric technology for providing QoS. Differentiated Services is a Layer 3 traffic handling mechanism and it applies only to the Internet Protocol (IP). In a typical DiffServ application, sending entities mark each transmitted packet with the appropriate bits in the Layer 3 header of the IP packets. Routers within a DiffServ network use the DSCP markings to classify packets and deal with using the appropriate queuing mechanism they use. Since this packet classification and queuing is done independently at each router DiffServ becomes a per-hop behavior (PHB) mechanism. Layer 2 switches and routers may also recognize DSCPs and support the corresponding per-hop behavior. A PHB is the behavior applied at each forwarding node and it alone is insufficient to define a complete end-to-end QoS guarantee but by concatenating a group of nodes supporting PHB, it is possible to define a complete end-to-end service guarantee. An example could be a group of router nodes along a pre-specified route supporting the Expedited Forwarding (EF) (type of) PHB could yield a suitable service

for voice and video conference. Another example of a PHB besides EF, where packets are transmitted from the network ingress to the egress with minimum delay, is the Assured Forwarding (AF) group where preference is given to the guaranteed delivery of the packets by the router queuing system with no particular emphasis to latency.

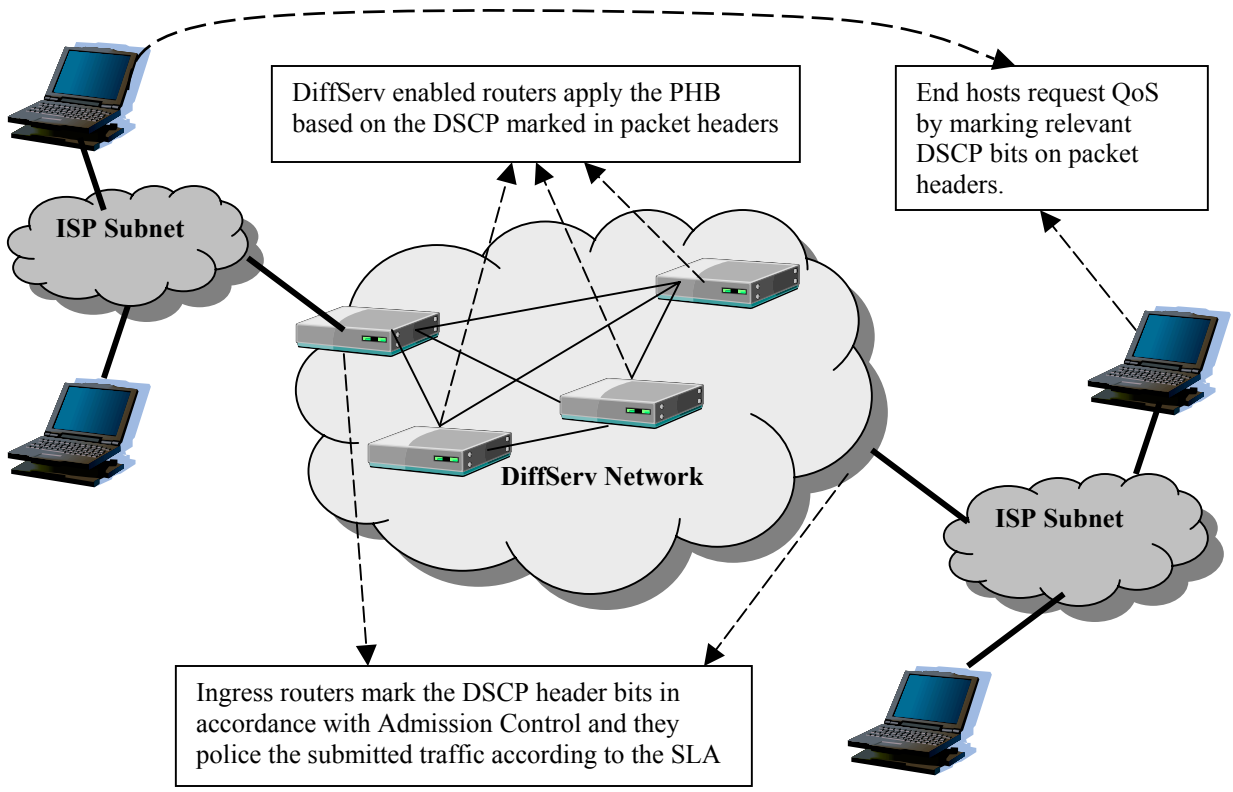


Figure 2.4: Concepts of a DiffServ Network

Service parameters are specified at edges of the DiffServ network in the form of Service Level Agreements (SLAs) that define the amount and type of customer traffic entitled to pass through a specified QoS level. An **Admission Control Service** (ACS) is the process through which any traffic is admitted to the network and allotted a reservation level depending on user authentication, authorization and resource availability. The SLAs are validated by the ACS agents run on chosen nodes of a network. More than one ACS agents are chosen by an election system. DiffServ being a packet marking QoS mechanism marks a particular field called ***DiffServ codepoint*** (DSCP) in the IP header of

the packet to be transmitted. This 6-bit DSCP field spans the fields formerly known as the *Type-of-Service* (TOS) field and the IP precedence field. These fields have been limited in their use since many years to provide a form of DiffServ. For instance, the IP precedence field has been used by network equipment to identify important network control traffic so that it is dealt with preferentially. A simple diagram in figure 1.5 shows the operating concepts of a typical DiffServ network as discussed above.

Integrated Services (IntServ)

The Integrated Services working group of the IETF was formed in 1995 and was chartered with the definitions of the services to be provided by IP network elements. It focuses heavily on multimedia traffic on the Internet and the traffic handling mechanisms it deals with provide very quantifiable and measurable service characteristics. IntServ does not necessarily assume per conversion (or per flow) traffic handling but it has primarily evolved being associated with the RSVP signaling protocol, which is a per-flow signaling protocol. It will be described in detail in a following section. This per-flow model of IntServ contrasts starkly with the aggregate traffic handling mechanisms of provided by 802 user_priority and DiffServ. Presently, two services are defined within the IntServ framework: the *Guaranteed Service* and the *Controlled Load Service*. The Guaranteed Service aims at carrying a traffic flow under quantifiable latency bounds and the Controlled Load service promises to carry a related traffic flow with the appearance of a lightly loaded network. These are quantifiable services because the QoS parameters such as bandwidth and latency can be numerically specified. On the other hand certain qualitative DiffServ services' parameters may not be specifiable.

Integrated Services over Slow Links (ISSLOW) fragments IP packets at the link layer using standard PPP multi-link fragmentation for transmission over slow link such as dialup connections such that any fragment does not occupy a link for longer than some threshold value. This ensures reduced latency for competing traffic. Most of the work presented in this document uses the Resource Reservation Protocol (RSVP) for carrying out analysis and implementation on a test bed. RSVP is a part of the IntServ traffic handling mechanism. In the following section RSVP is described in detail.

2.4 Integrated Services and Resource Reservation Protocol

The IntServ model needs an implementation framework to support the services it defines. This framework consists of the following four components

1. Packet Scheduler

The packet scheduler is the component that sorts out the traffic in a network node according to defined queuing algorithms such as FIFO, fair queuing, weighted fair queuing etcetera. It ensures that any legitimate flow is neither starved of nor surfeited with network resources.

2. Packet Classifier

This module makes sure that arriving packets are assigned to relevant queues holding priorities in congruence with the service level agreements.

3. Admission Control

Admission control is a network element or process that decides whether sufficient resources are available to accommodate a new service request without affecting the already admitted service requests. This dynamic negotiation process occurs each time that a host requests some real-time service or an explicit admission control may be made static by requesting resource reservation in anticipation of traffic arrival on the network. When *explicit* admission control is exercised, the network may admit or reject the flow explicitly and when implicit admission control is exercised, there is no analytic agreement between the network elements or the requesting application. Instead, the senders submit packets and the network element carries them if available resources suffice or *police* them (accept or discard based on policy) if otherwise.

4. Reservation Setup Protocol

According to the authors of RFC 2213, “there is an inescapable requirement for routers to be able to *reserve* resources.” and that “an explicit setup mechanism is necessary.” These

requirements led to the development of the Resource Reservation Protocol or RSVP as the reservation setup protocol mentioned in the implementation framework.

RSVP was developed to be a signaling protocol for establishing simplex reservations across routers. It was also realized to use a *soft-state* model so that the reservations would expire after certain time if not explicitly periodically renewed. RSVP is a receiver-oriented protocol in the sense that resources are not actually reserved until receiver signaling happens in subsequence to the sender signaling. RSVP inherently supports multicasting applications by defining merging reservations rules and not affecting the multicast scalability. The end-to-end philosophy of RSVP guarantees that the resource reservation does not rely upon the maintenance of the state by the underlying network and this state may be destroyed when the endpoint applications cease to exist themselves. A RSVP *session* contains information about each unicast or multicast flow. Bi-directional flows need two unidirectional sessions to be set up.

2.4.1 RSVP Messages and Basic Protocol Operation

RSVP is based on a set of message exchanges between the sender and receiver. RSVP messages carry a list of objects, each of which specifies a set of related information. The fundamental messages are the PATH and RESV messages. The senders use the PATH messages to set up a *downstream* route to the receiver and recommend quantitative reservation parameters for the data exchange session. The receivers reply with the RESV messages that travel *upstream* along the route setup by the PATH messages and confirming the parameters to setup the actual resources. This message exchange happens once every few seconds to refresh the RSVP soft state or else the reserved resources would be freed if the reservation state expires. Figure 2.5 below shows the process of reservation that happens when RSVP is used and the cycle of events that takes place.

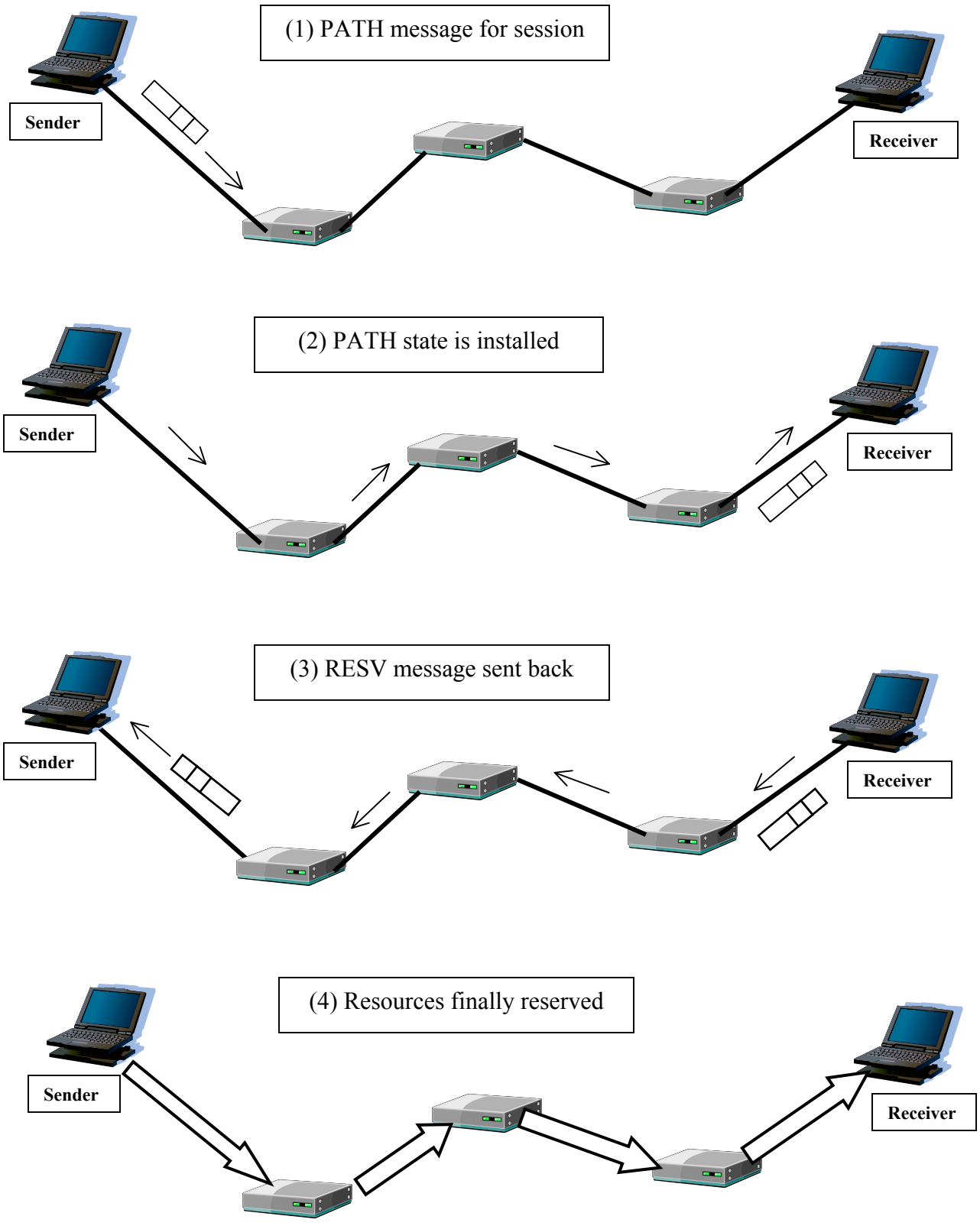


Figure 2.5: Reservation cycle

PATH Messages

The RSVP reservation process initiates with the transmission of periodic PATH messages by the sender application. The PATH messages have the following functionality:

- Sets up a definite route for the downstream flow towards the receiver
- Defines the session for the RSVP session
- Describes the traffic characteristics in detail
- Maintain the soft state alive by periodic transmission (REFRESH messages)

RSVP HEADER
SESSION object
RSVP_HOP object
TIME_VALUES object
POLICY_DATA object
SENDER_TEMPLATE object
SENDER_TSPEC object
ADSPEC object

Figure 2.6: Structure of a PATH message

The figure 2.6 above illustrates the composition of an rsvp path message and the associated objects. Further description of the objects is as follows:

RSVP header: is the standard header placed at the beginning of each PATH message

SESSION object: identifies the RSVP session to which the sender is sending. The triplet of receiver IP address, port and protocol number defines the session.

RSVP_HOP object: contains the PHOP (previous hop) and the NHOP (next hop) objects that are used by RSVP aware devices in the route to specify and store the path information used by the flow and the subsequent RESV messages.

TIME_VALUE object: contains the refresh time interval for periodic exchange of PATH and RESV messages for maintaining the soft state of the RSVP session.

POLICY_DATA object: contains policy data for the traffic generated by the sender application to assist in session authentication and admission.

SENDER_TEMPLATE object: identifies the sender of the flow session

SENDER_TSPEC object: contains the traffic specs as suggested by the sender for the described flow

ADSPEC object: contains accumulated resource information collected en route from sender to receiver to reflect the types of service supported, available bandwidth, latency, maximum packet size and information about number of devices being RSVP aware

The primary purpose of the PATH message is to set a bi-directional path for the traffic flow. In general it cannot be guaranteed that the reverse route will follow the same path as the forward route in any data flow. Using the RSVP_HOP objects, the PATH messages offer a mechanism for en route RSVP aware devices to copy and store the PHOP information and write its own address (IP or MAC depending on the device's layer) in the PHOP field before forwarding the packet. This ensures that each router/switch will remember the device that sent it the PATH message and thus sends the generated RESV or data messages on the exact same route upstream.

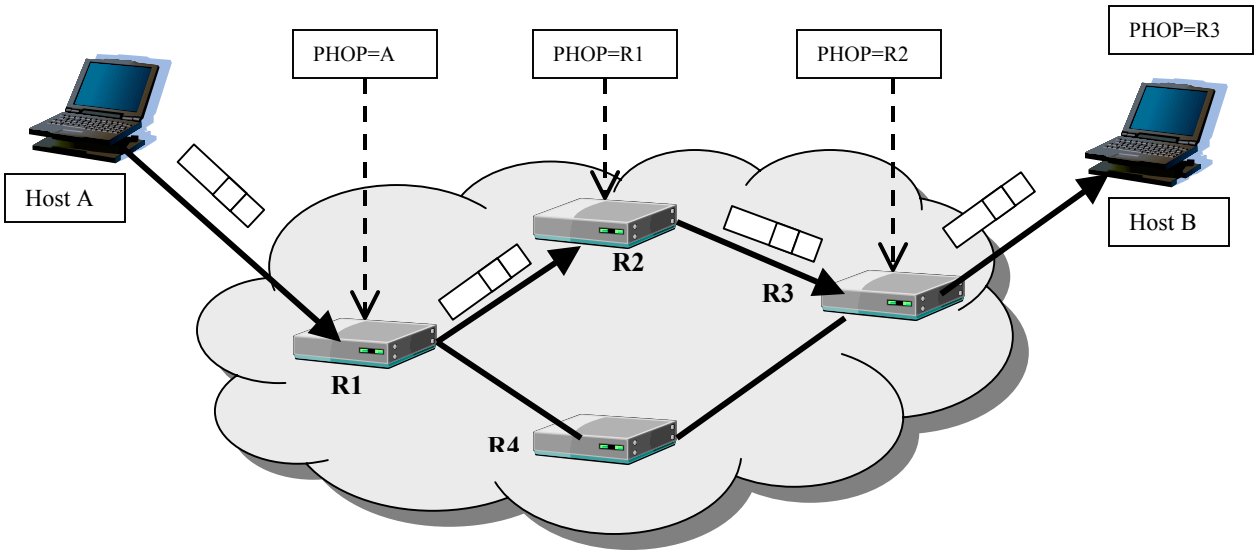


Figure 2.7: PATH messages mark the bi-directional route for RSVP traffic flow

As seen in figure 2.7, the PATH messages set a fixed downstream route A-R1-R2-R3-B for bi-directional data flow as a result of which reverse RESV or any data packets follow the route B-R3-R2-R1-A and do not traverse through R4 even though it is a valid option.

RESV Messages

After the sender establishes the session and route information on the routers via the PATH messages, the receiver issue RESV messages to corroborate their interest in receiving the flow with certain quantifiable flow reservations.

The functions of a RESV message are as follows:

- It specifies the required type of service (for example, Guaranteed, Controlled Load, Null, or others).
- Incase if quantitative service, it specifies the quantity of resources that need to be reserved expressed in terms of the FLOWSPEC objects.
- It specifies one of more senders and a reservation style such as Fixed Filter, Shared Explicit and Wildcard Filter.

RSVP HEADER
SESSION object
RSVP_HOP object
TIME_VALUES object
RESV_CONFIRM object
POLICY_DATA object
STYLE object
FLOWSPEC object
FILTER_SPEC object

Figure 2.8: Structure of a RESV message

As seen in figure 2.8, the RESV message is structured not too dissimilar from the PATH message with the four different fields described below:

RESV_CONFIRM object: is an optional object requiring the sender to send a confirmation of the reservation
STYLE object: is used to specify how reserved resources are to be allocated incase of a multicast session with multiple senders.
FLOWSPEC object: carries the detailed description of the resource reservation demanded by the flow. This object is based on the popular token bucket model.
FILTER_SPEC object: carries the subset of senders from which traffic is to be protected.

As the RESV messages travel upstream towards the sender, they are analyzed by each network device they pass through. The quantity of requested resources and the requested service type are compared against the amount of resources available on the device. If the resources are available then a RESV state is installed or else an RESV_ERR message is sent downstream to the receiver. This admission control process is handled by an entity called the Subnet Bandwidth Manager (SBM) that is described in detail in a later section.

2.4.2 Removing Reservations

Routers maintain the reservation soft state until they time out due to a delay in refresh messages exchanged. Alternatively the reserved resources are released if either the sender of receiver sends an explicit TEAR message over the route. Senders would send the PATH_TEAR message and receivers would send the RESV_TEAR message. These messages are not complimented or reciprocated by other host. Instead of sending the tear messages, another way applications may terminate their reservation is simply by ceasing to send the messages. This, however, would be a wasteful method, as network devices would carry the reservations at least till the timeout interval. Typically, besides sending the tear messages at the session end, applications tear down the reservations when they encounter a RSVP error message such as PATH_ERROR or RESV_ERROR.

2.5 Subnet Bandwidth Manager

2.5.1 Improving the quality-efficiency of a network

The most important tradeoff of implementing Quality of Service on a network is network efficiency or utilization. This tradeoff is quite aptly expressed as the *quality-efficiency product* [BER]. As the service guarantees for providing QoS improve , the efficiency of the network suffers due to possible over provisioning of resources. For example, as the number of privileged voice traffic circuits in a system increase (i.e. network efficiency increases), the latency encountered by the average packet increases(i.e. service quality decreases). This effect may not be prominent over a general Internet route containing backbone routers and gigabit networks but it certainly is significant when operated over Ethernet or an IEEE 802-style LAN technology.

QoS in 802-based networks is supported using *user_priority* values and traffic classes. Layer 3 devices mark the packets with the appropriate *user_priority* values by appending an *802.1Q* header to each transmitted packet. Alternately, layer 2 ingress devices may append a certain user_priority value based on classification information or the packets submitted to Layer 2 already have been marked with this user_priority value. The priority values are mapped to different traffic classes [IEEE8021P]. This is similar to the DiffServ approach that uses DSCP markings. Since the effort must always be to increase the quality-efficiency product, a mechanism is needed to prevent a network from being over-provisioned in terms of resources. This may be achieved by restricting new users demanding guaranteed service by using some sort of explicit admission control.

To improve the quality-efficiency product it is necessary to look beyond the simple usage of *user_priority* as briefly described above. IETF recommended a more sophisticated signaling approach for devices in 802-style networks, which is similar to the RSVP signaling. The Subnet Bandwidth Manager (SBM) has been defined as an admission control agent over IEEE 802-style networks. SBM provides a method for mapping an internet-level reservation setup protocol such as RSVP onto IEEE 802-style networks. It

describes the behavior of RSVP-enabled Layer 2, 3 and host devices to reject or grant LAN resources when requested by RSVP enabled data flows.

2.5.2 SBM and SBM Protocol

As defined in RFC 2814, the Subnet Bandwidth Manager (SBM) is some protocol entity that exists in each network segment and assumes the responsibility for admission control for the resource reservation requests originating from clients attached to that segment. In case of multiple SBMs existing in a network segment, actual admission control is done by one *Designated SBM* (DSMB) selected by means of an election algorithm. The presence of a DSBM makes the segment a *managed segment*. It is possible for some Layer 2 LAN segments to be interconnected by devices that are transparent to the SBM protocol. In such cases they may still be managed by a DSBM that does not need to be physically present on the LAN segments it manages.

Besides being an admission control module, RFC 2814 also defines SBM as a signaling protocol for RSVP based admission control over 802.xx type of networks. By means of this signaling protocol an SBM performs the following tasks

1. Announce its presence and SBM capabilities on the network
2. Participate in the DSBM election (incase multiple SBMs are present)
3. Announce itself as the DSBM in case it is elected as one
4. Receive and forward RSVP messages via the multicast DSBM address
5. Maintain policies regarding admission control and resource allocation
6. Maintain the reservation state for each admitted flow
7. Ensure resource management across Layer 2, Layer 3 and other SBM transparent devices

Clients or hosts on a network are informed of the presence of an elected DSBM by periodic transmission of ***I_AM_DSBM*** messages sent to a broadcast address 224.0.0.17 known as *AllSBMAddress*. These clients become DSBM clients because all RSVP signaling messages would now be sent through the DSBM and not directly to the peer.

This would enable the DSBM to implement policy control looking at the source/destination IP address. The DSBM clients would initiate RSVP signaling by sending PATH messages to the multicast address 224.0.0.16 known as the *DSBMLogicalAddress*. DSBMs monitor this address to detect PATH messages sent to it and forward the PATH messages to the *AllSBMAddress* incase the next hop is another DSBM. Alternatively, if the next hop is not a DSBM (i.e. the forwarding segment is not managed) the PATH message is forwarded to the *DSBMLogicalAddress*.

Here is a table referred from RFC 2814 that elucidates the use of the two multicast addresses used by the SBM protocol

Table 2.1: Functionality of DSBM multicast addresses

Type	**DSBMLogicalAddress 224.0.0.16**	**AllSBMAddress 224.0.0.17**
DSBM Clients (senders or receivers)	• Send path messages to this address	• Monitors this address to receive PATH messages forwarded by the DBSM • Monitors this address to detect the presence of a DBSM
SBM (not designated)	• Send path messages to this address	• Monitors and sends on this address to participate in election of the DSBM • Monitors this address to receive path messages forwarded by the DSBM
DSBM	• Monitors this address for PATH messages directed to it	• Monitors and sends in this to participate in election of the DSBM • Sends path messages to this address

2.5.3 Differences from RSVP

SBM protocol is mostly similar to RSVP in terms of its message processing and forwarding nature. The PATH message contains a PHOP (Previous Hop) object that contains the IP address of each forwarding entity (typically a Layer 3 device such as a router). This enables a static route to be set from the sender to receiver so that the corresponding RESV messages also follow the same path upstream to confirm resource reservation and flow information. As for the difference, consider a situation where, instead of a Layer 3 device, an IEEE 802-style network is a transient subnet in the route of a PATH message. A typical RSVP message contains the LAN_NHOP object that points towards the IP address of the next router en route. It is quite possible that some Layer 2 device may be designated as the SBM for that network segment. In this situation it is not reasonable to expect these devices to look in to the Layer-3 header to do an ARP (Address Resolution Protocol) translation to obtain the IP address of the next hop.

Therefore it is required that the LAN_NHOP object should contain not only the IP but also the Layer-2 MAC address of the next Layer-3 hop to forward the RSVP message to. SBM achieves this by introducing two parts in the LAN_NHOP object: LAN_NHOP_L2 and LAN_NHOP_L3 denoting addresses of Layer 2 and Layer 3 respectively. The LAN_NHOP_L3 address is used on L3 devices to forward the PATH messages towards its destination whereas the LAN_NHOP_L2 address by SBM protocol entities on a L-2 domain to determine the correct way of forwarding the PATH messages to its next L-3 hop destination which would be the egress point from the L-2 domain.

2.5.4 Similarities with RSVP

The DSBM processes the PATH messages flowing through it in the same way as specified in the RSVP message processing rules. It retrieves the IP address of the previous hop (PHOP) and stores it in its PATH state before forwarding the message with its own address as the PHOP address to the next hop. All RSVP related messages (such as PATH, PATH_TEAR, RESV, RESV_CONF, RESV_TEAR, and RESV_ERR) flow through the DSBM as it inserts itself as a hop between two RSVP nodes.

2.6 Pertinence of DSBM to this thesis

The results presented later rely heavily on flows being subject to admission control through a module that keeps track of the total resources available for reservation and those already committed to flows. A new mechanism is proposed called the Adaptive Buffer Window (ABW) mechanism that enables flows to adapt to the available bandwidth in case the current reservation is denied by the DSBM. Furthermore the DSBM keeps track of the number of flows admitted and the bandwidth allocated to each of them. Without a bandwidth manager it would be very hard to tweak the network test bed in terms of setting a cap for cumulative bandwidth reservation rate, cumulative burst rate and cumulative peak reservation rate. These parameters are fundamentally related to the token bucket model used by the RSVP protocol.

CHAPTER 3 Theoretical Analysis

MPEG encoded video carries a constant number of frames per second. Since the frame size varies constantly, this translates into a highly variable bit rate per second. Since streaming video is delay sensitive, QoS mechanisms must be used to guarantee playback quality even at a trade-off of over-provisioning of network resources. These aspects are described in this chapter.

3.1 Characteristics of VBR video

Variable-Bit-Rate (VBR) compressed video, such as MPEG video [DLG91], can exhibit significant bit rate variations over multiple time scales. In addition to the small-scale variations within the Group of Pictures (GOP) a video's bit rate is heavily influenced by the scene activity it contains. As compared to Constant Bit Rate (CBR) streams, VBR video is more efficient to users and service providers because it provides a better quality for the same average bandwidth used [ROS97]. If we stick to transmitting video at 30 frames per second (fps), each second those 30 frames worth of (MPEG-1 encoded) data may vary from approximately120Kbytes till approximately 600Kbytes depending on the type of video (for example, action, sports, talking-head etcetera) and the level of scene activity in the current play sequence. For MPEG-2 encoded videos with voice and high resolution each second's worth of data can be up to 1500Kbytes in size. A video with frequent scene changes would increase the average frame size in a GOP for the duration of intense scene activity. This is attributed to reduced inter-frame correlation and degree of temporal prediction between frames. Thus the variation in data rate is not only visible in different genres of videos but also within different sections of the same video.

This bursty nature of VBR video sources poses some important issues for video delivery over high-speed networks. On one hand, we may want to provide satisfactory and guaranteed levels of QoS for video traffic and this is usually done by allocating network bandwidth at or close to the peak rate of the entire transmitted video. On the other hand, we want to utilize network resources efficiently considering the fact that reserving a static

bandwidth level would lead to over provisioning of network resources. For any bursty video source the peak rate would be significantly larger than its average rate and bandwidth allocation at peak rate would be wasteful for resources. Figure 3.1 displays the burstiness of MPEG traffic as plotted using the traffic trace from a video clip 'News' obtained from [ROS97].

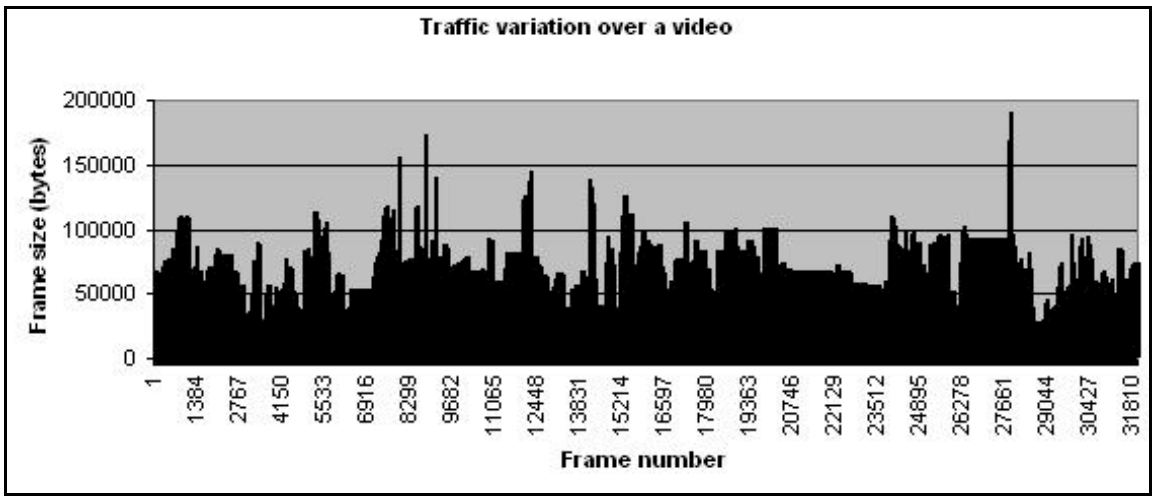


Figure 3.1: Distribution of traffic over the playback length of a typical video

The typical problems associated with transfer of bursty portions of VBR data include buffer overflow on the receiving client's side and unexpected packet loss or latency that make the playback jittery and of inferior quality. In an under loaded network burstiness is absorbed but in a more commercial setup where the video streams share common links, burstiness causes congestion problems as the number of VoD clients grow. Various mechanisms have been suggested in the past to improve network efficiency and performance. Some of them worth mentioning are:

Stream tapping [CLP00]: which allows clients to "tap" into streams of data created for other clients who have requested the same video. By using existing streams as much as possible, clients minimize the amount of new bandwidth they require and allow more clients to simultaneously use the VOD server.

Optimal buffering and smoothing [FJS97**,** SZ96]: where client buffer space is fully utilized to fix a finite bandwidth reservation level for the entire movie duration. This causes a large start up delay but assists in admission control since the network sees one constant bandwidth value for the entire playback duration. Any admission control module such as an SBM can safely ignore the small time scale variation in bit rate.

Optimal Scheduling / Statistical Multiplexing [KT97]: exploits the temporal structure of pre-recorded MPEG video to achieve bandwidth gain via statistical multiplexing and optimal scheduling of video streams. By introducing a negligible startup delay, this technique yields significant bandwidth savings.

Resolution Adaptation [KK03]: uses the object-based approach for MPEG-4 encoded videos to dynamically drop the resolution if bandwidth is insufficient. Similar work also exists for MPEG-1 encoded videos.

Most of these approaches work against the static allocation of network resources for VBR video and suggest work around ways to achieve efficiency. Looking at the strong need to change the static reservation level paradigm, **Renegotiated Constant-Bit-Rate** (RCBR) and Piecewise CBR (PCBR) mechanisms have been suggested [MSD97, CK96]. The RCBR mechanism tries to address the presence of burstiness of VBR video over multiple time scales by allocating different amounts of bandwidth for different segments of a video. It does that looking at the traffic pattern file of the video that contains a distribution of frame sizes over the duration of the movie. This aims at providing the required QoS guarantees without over provisioning of network resources. Since all video traffic entering the network is always under a CBR constraint, RCBR can simplify the requirement on buffering and scheduling in network switches compared to VBR service that allows traffic bursts into the network. RCBR can be used either for pre-recorded videos the traffic profiles of which are already available or for live videos where online traffic prediction is employed.

As a preliminary analysis study and motivation for carrying out experiments on the test bed network we analyzed several real world MPEG encoded traces obtained from [ROS97] for their bandwidth saving potential when RCBR mechanism is used instead of traditional static CBR reservation level. We show that even simply using fixed renegotiation intervals, huge savings on network bandwidth can be achieved without sacrificing QoS. We then consider a framework of VoD based on dynamic reallocation approach with a new concept of video traffic pattern server system. As an example the plots shown here belong to a news show that would come under the moderate bursty category.

3.2 Current implementation of QoS for VBR video

Out of VBR, CBR and ABR (Available Bit Rate scheme used in ATM networks) a CBR data stream is the easiest to deal with. In order to assure perfect bandwidth usage the average bandwidth of the video can be used as the allocated bandwidth and all the burstiness could be absorbed using large buffers. For the MPEG-1 compressed movie 'Starwars' the average data rate is 0.6Mbps and to use this as the allocated bandwidth a receiver must have a 22Mbyte buffer; and for the I-frame-only compressed version having a constant rate of 6.6Mbps, the receiver needs a 93Mbyte buffer [MWG93]. Certain buffering mechanisms have been suggested (such as [SZ96]) that avoid buffer overflow and starvation with the tradeoff of additional delay. However, the pre-fetch delay introduced by buffering may not be suitable for delay sensitive real time video streams.

For premium quality video transmission, QoS schemes such as RSVP or MPLS are implemented to guarantee loss less data transfer with minimum latency. RSVP is a protocol used to reserve certain amounts of bandwidth on intermediate routers between the video server and clients. The main weakness of RSVP for wide-scale deployment is its per flow 'stateful' behavior which does not scale well with increasing the number of clients, as RSVP requires each QoS aware device on a route to maintain some state information for each flow. This restriction is overcome by the Differentiated Services

technology that marks packets belonging to prioritized flows and uses protocols such as the MPLS (Multi Protocol Label Switching). MPLS deals with a group or a class of flows rather than each flow independently. This removes the RSVP overheads of refresh updates and maintaining state information for individual flows.

Conventionally the RSVP protocol is used with a static reservation level that most often corresponds to the bandwidth required by the most bursty part of the video stream. This ensures a congestion free network as seen by any playback portion of the data stream. RSVP chooses a constant token rate (since it uses the token bucket scheduling model) that represents the reserved bandwidth in bytes per second for the entire playback duration of the video. The most important advantage of choosing a constant token rate is the ease in admission control for the flow. For a variable bandwidth reservation level such as RCBR it would take a lot more intelligence on the admission control module to decide whether statistical multiplexing of the incoming stream with the rest of flows would not cross the total bandwidth threshold.

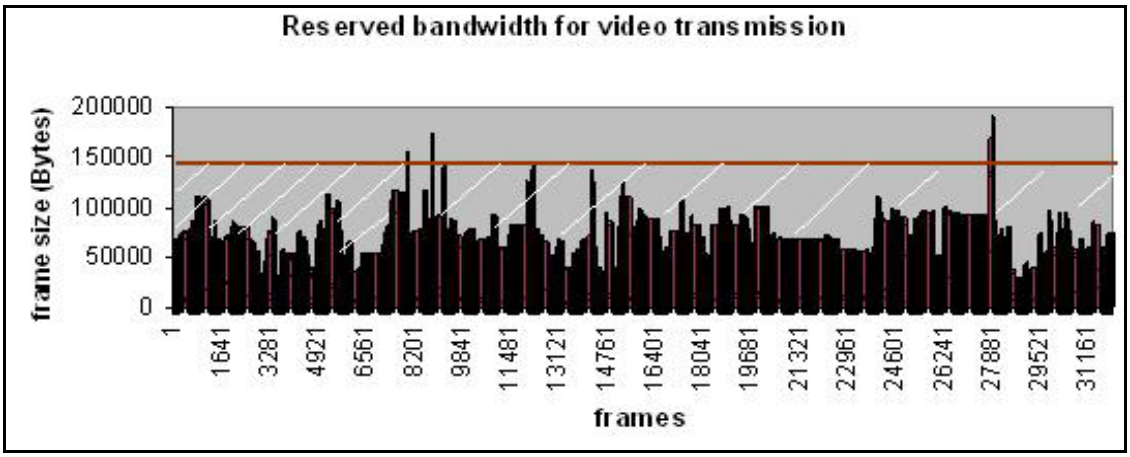


Figure 3.2: Solid line depicts the constant reservation level for video file of Figure3.1

As can be seen from Figure 3.2 above, using a static bandwidth allocation scheme such as CBR, a constant bandwidth is allocated during the call setup for the entire video session. The shaded region in the figure represents the unutilized network resources that have been reserved for the traffic stream. For a bursty action movie such as Terminator this

shaded region totals to about 3 Megabytes for a 30-minute movie sample (which is almost 10% of the file size). In contrast, using the RCBR scheme, a video transmission rate can be dynamically negotiated and bandwidth can be re-allocated accordingly.

3.2.1 Concept of Traffic Windows

Now consider dividing the MPEG video trace shown above into multiple windows of time such that each window contains a fixed number of MPEG frames for bandwidth renegotiation. The size of the window (expressed in frames) can either be fixed or varied in RCBR during the entire video transmission session. In this research work we shall focus our analysis on the fixed window scheme due to its simplicity in the dynamic bandwidth re-allocation mechanism and its compatibility with the RSVP REFRESH message timing as will be discussed shortly. If we can change the bandwidth reservation each time we have a new window to transmit, significant savings in reserved bandwidth can be obtained as compared to the conventional method of reserving a very large static bandwidth for the entire duration of the video. The QoS experienced by the data stream packets would be just as effective in both cases because the peak data rate in each window is still equally accommodated by the bandwidth reservation for that particular window.

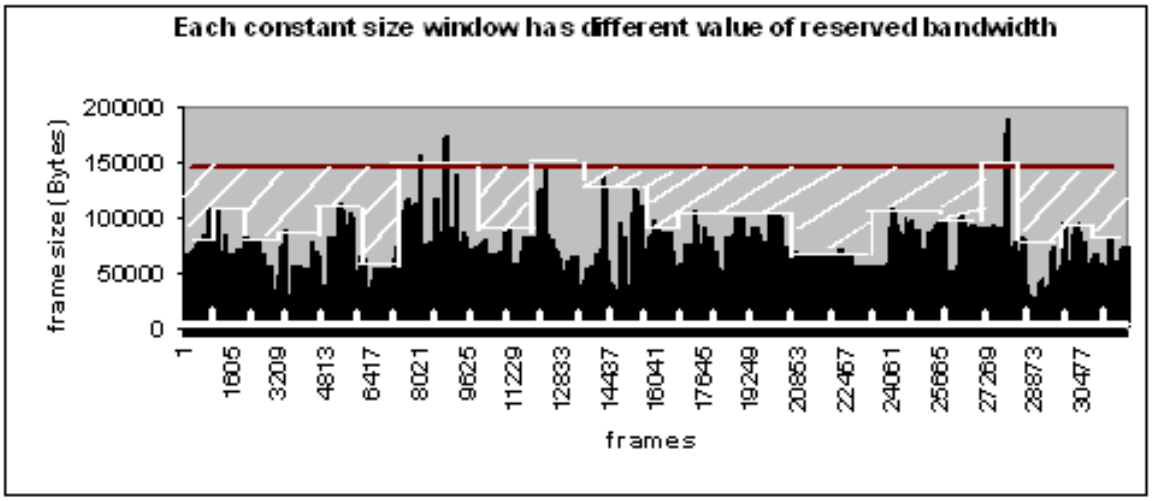


Figure 3.3: Shaded region is savings in reserved bandwidth.

Figure 3.3 above depicts dynamic bandwidth re-allocation levels for each window for the video trace shown in Figure 3.1. Each of the windows above could be hundreds or even thousands of frames wide depending on the video's traffic profile and the RSVP refresh message interval chosen. As seen in the plot, the solid lines enveloping the peaks are the reservation levels for each window and the shaded area with diagonal white lines is the bandwidth reservation difference between the conventional static CBR and the RCBR. This area represents the saving of link bandwidth, which directly implies that we can accommodate more flows onto a single link if we follow this scheme. The best thing about sending pre-recorded video is that we already know its traffic characteristics in advance in the form of traffic pattern file that tells us the size of each frame in the video. MPEG encoders (and certain other software) can help us to generate such traffic pattern file from a given video file. Once the window size and the renegotiation rates for bandwidth re-allocation are decided and computed offline for a pre-recorded video trace, RSVP can be used for rate adaptation.

RSVP creates a flow-specific reservation 'soft state' at each router between the server and the client, which needs to be kept alive by sending periodic RSVP REFRESH messages. The typical and recommended refresh interval is 30 seconds. The '***flowspec***' data structure (which is a part of PATH and RESV messages) specifies flow and reservation parameters related to token bucket model for the video stream. Rate adaptation through the RCBR scheme would either require sending explicit PATH/RESV signaling messages at the time of bandwidth renegotiation or, more efficiently, piggy-backing the periodic refresh messages with updated flowspec parameter values. In the latter case no extra overhead would incur as these flowspec packets in RSVP are anyways sent periodically for state management, if the refreshment period for rate adaptation is a multiple of that for state management.

Despite the existing concept of RCBR, bandwidth updates in conventional systems were not implemented via RSVP REFRESH messages, which were seen more as QoS control messages rather than means to update the reservation level.

3.3 Analysis of RCBR

In this section we aim to show the gains in network utilization and efficiency using the RCBR mechanism for real-world VBR video traces obtained from [ROS97]. In our analysis we compare the static and dynamic bandwidth reservation schemes to compute the following two measures:

(1) Average bandwidth saved, and
(2) Relative bandwidth efficiency.

Let r_i (i = 0, 1, 2, N-1) denote the actual average video transmission rate for time window *i*, and let $BW_{RCBR(i)}$ denote the bandwidth allocated for time window *i*. Let BW_{CBR} denote the bandwidth allocated for the entire video transmission session, and BW_{saved} denote the average bandwidth saved after using RCBR dynamic bandwidth reservation compared to conventional static CBR reservation. For the sake of generality, we introduce a parameter α ($0 < \alpha \leq 1$) to express the assumed reservation level as a fraction of the peak rate in transmission duration under consideration (either a time window or the entire transmission session). Thus any bandwidth reservation level can be expressed as a function of *α*, in general. We can express the two measure of RCBR performance as the following mathematical expressions:

$$BW_{saved}(\alpha) = \mathrm{BW}_{\mathrm{CBR}}(\alpha) - \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=0}^{\mathrm{N-1}} \mathrm{BW}_{\mathrm{RCBR}}(\mathrm{i}, \alpha) , \qquad \ldots(1)$$

$\mathrm{BW}_{\mathrm{saved}}$ represents the bandwidth savings incurred when using RCBR instead of CBR for the playback duration of a movie. This quantifies the shaded region shown in Figure 3.3. Bandwidth saved is an absolute measure indicating the over-provisioning of resources on the network. It increases with the increase in α because a higher α implies more bandwidth is being reserved for the data stream. It decreases with window size because a larger window of frames will statistically have a higher peak – average ratio and

reserving bandwidth at higher peak rate would prove more wastefulness for the same amount of data bytes.

The relative bandwidth efficiency, η, is a ratio defined in Equation (2), which estimates the degree of how much bandwidth we can save by using fixed window size RCBR as compared to static CBR for bandwidth reservation.

$$\eta(\alpha) = \frac{BW_{CBR}(\alpha) - \frac{1}{N}\sum_{i=0}^{N-1} BW_{RCBR}(i, \alpha)}{BW_{CBR}(\alpha) - \frac{1}{N}\sum_{i=0}^{N-1} r_i} \quad , \qquad \ldots(2)$$

where, $BW_{RCBR}(i, \alpha)$ = the CBR BW for the time window i,(i = 0, 1,2 ,…, N-1) and for a chosen value of α.

If the size of each time window (expressed as the number of frames) is chosen as one, there would be as many time windows as the number of frames in the video trace, and the actual bandwidth would be re-allocated for each frame. In terms of equation (2), $BW_{RCBR}(i, \alpha) = r_i$. Then $\eta(1) = 1$, which would be ideal but not practical due to too much overhead for the bandwidth re-allocation. On the other hand, when only one time window is used for the entire video trace, then RCBR is simply reduced to CBR, and thus $\eta(\alpha)=0$.

We analyze the performance of RCBR in terms of BW_{saved} and η for different values of RCBR window size and α. In our experiments, several real-world MPEG-I VBR video traces obtained from [ROS97] are used. Analytical results for different video traces with variable level of activity are presented in Figures 3.4 and 3.5. First, relatively calm videos with infrequent scene changes are considered. Second, the traffic traces with moderate scene activity are used to show the bandwidth savings and finally, highly bursty videos are used to show intense activity with a lot of scene changes. Higher screen activity generally implies greater burstiness for that MPEG video. Note that BW_{saved} indicates the shaded area as shown in Figure 3.3.

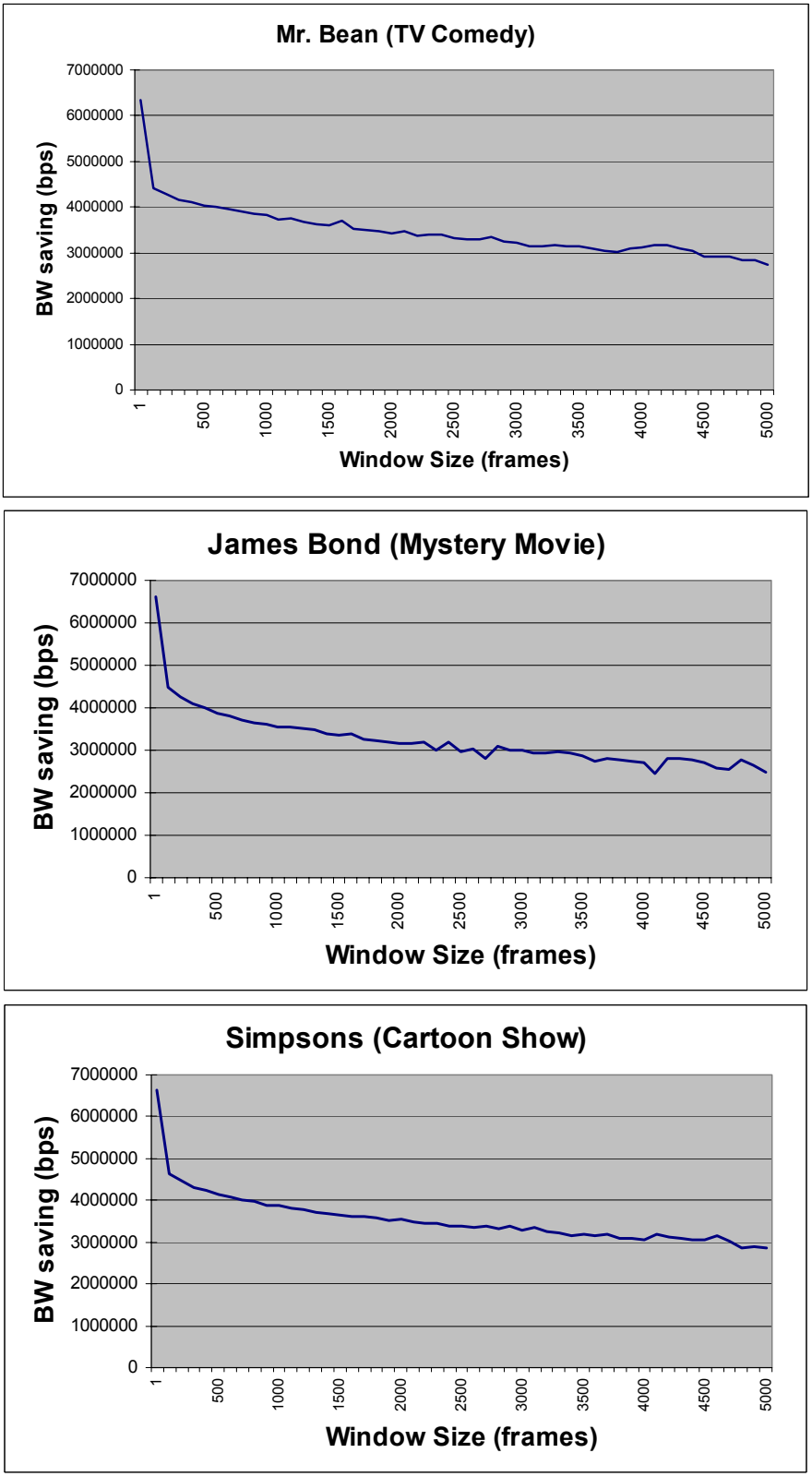


Figure 3.4. (a): Less bursty videos showing high gains from RCBR

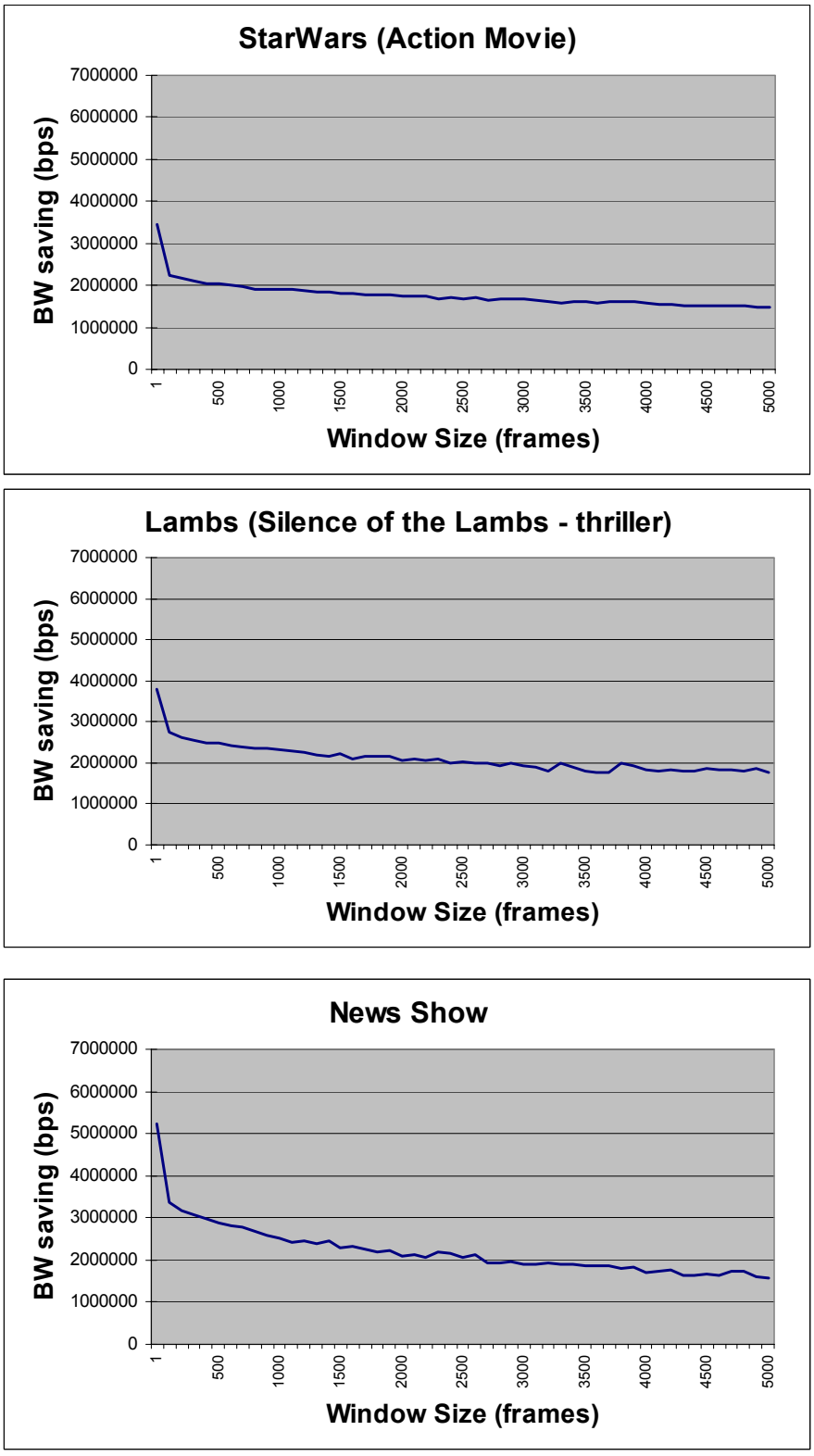


Figure 3.4.(b): Moderate bursty videos showing moderate gains from RCBR

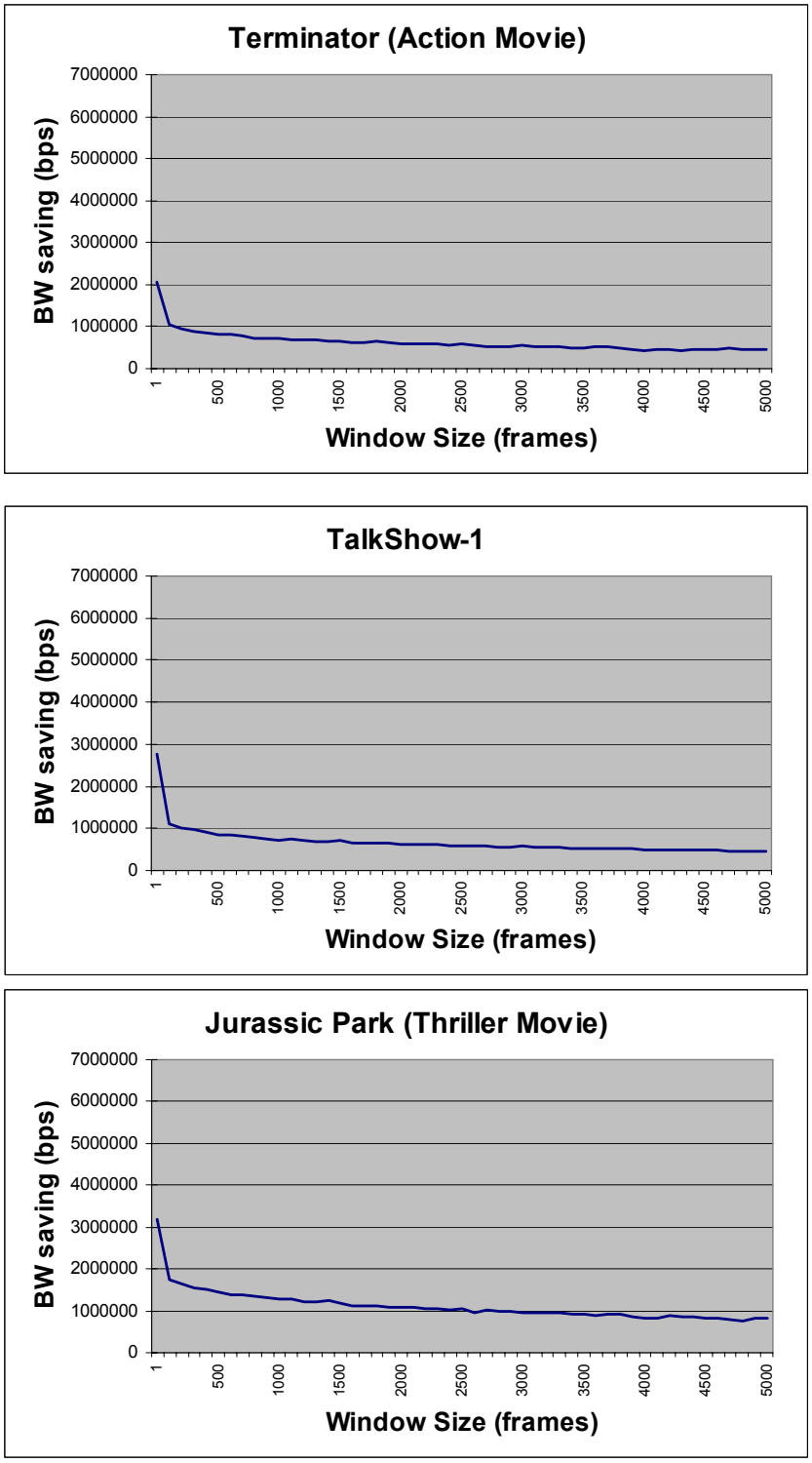


Figure 3.4.(c): Highly bursty videos showing low gains from RCBR

As seen in the figures 3.4 (a)-(c), with α = 1 and time window = 900 frames (corresponding to 30 seconds RSVP REFRESH period for a 30fps video), a less bursty movie such as 'Mr.Bean' or 'Simpsons' saves about 6Mbps of reserved bandwidth. These movies are characterized by infrequent scene changes and a calm traffic pattern. Moderately bursty videos like 'Starwars' incur between 3Mbps to 5Mbps of bandwidth savings. A highly bursty video such as 'Terminator' or 'Jurassic Park' saves about 1Mbps of reserved bandwidth. For less bursty videos high bandwidth efficiency can be achieved even using very large sizes of time window (*i.e.*, several thousand frames), while for highly bursty videos like 'Terminator,' the benefit of RCBR decreases quickly as the size of the time window for bandwidth renegotiation increases. The observations indicate that different time window sizes should be employed in RCBR mechanism for different burstiness levels of videos to be delivered. The reserved bandwidth savings can be compared with the peak and mean data rate of a file looking at Table 3.1. With multiple parallel flows the performance of a server with RCBR increases linearly. This trend is demonstrated empirically in chapter 5.

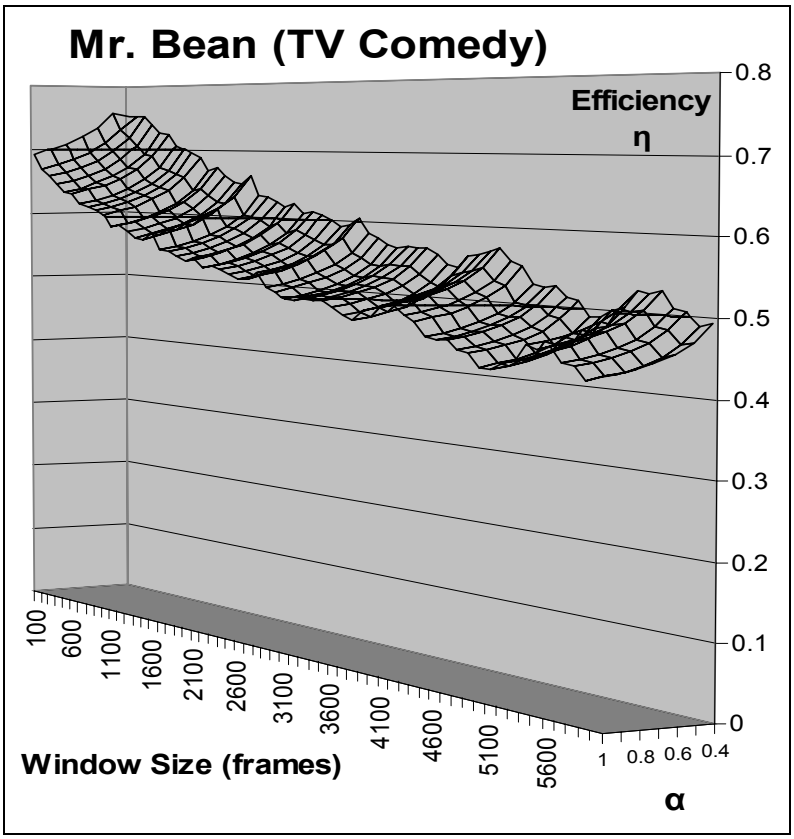
Table 3.1: Mean data rates for video traces of figure 3.4

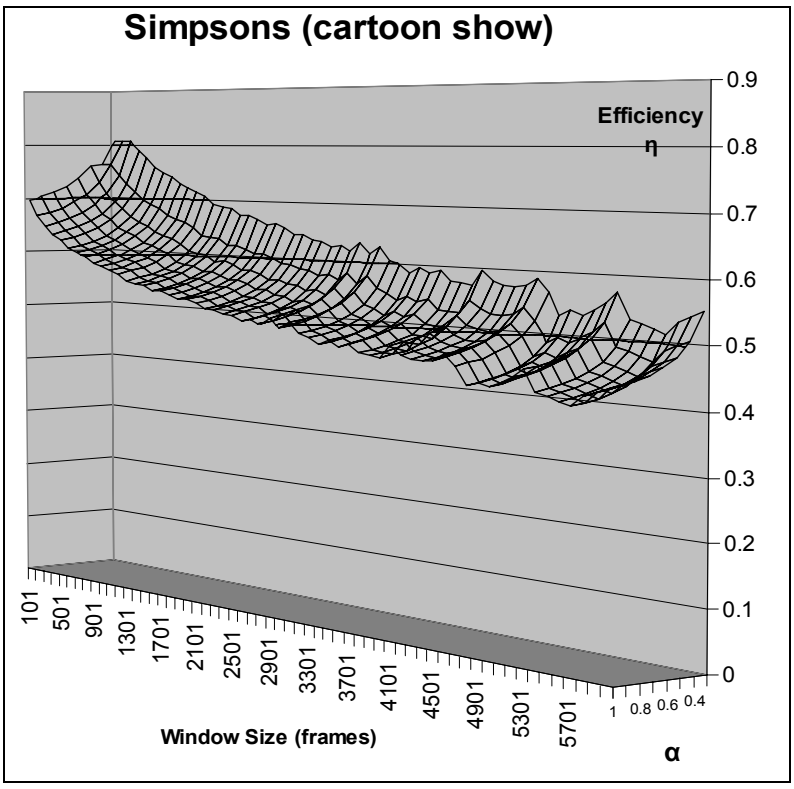
Name	**Mean Data Rate**	**Peak Data rate**
	Mbps	Mbps
Bond	0.729	7.337
Jurassic Park	0.392	3.588
Lambs	0.219	4.026
Mr.Bean	0.529	6.872
News	0.46	5.697
Simpsons	0.557	7.211
Starwars	0.279	3.744
TalkShow-1	0.436	3.203
Terminator	0.327	2.387

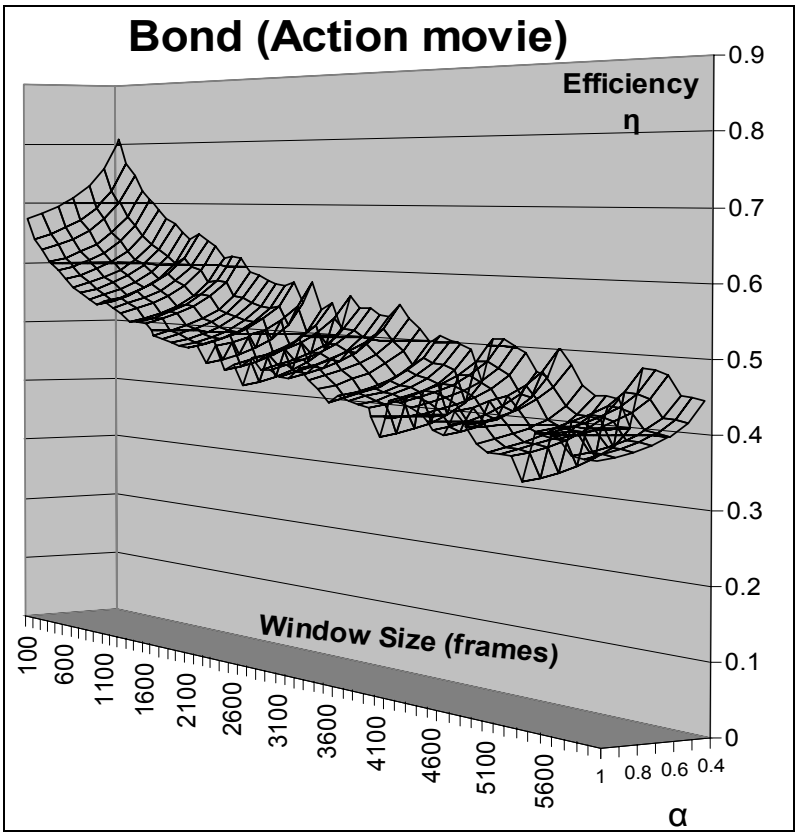
We computed these characteristics based on the frame traces obtained from [ROS97]. Although they were encoded using 25 frames per second (fps) playback rate, our calculations were done using a more generic value of 30fps. Since the trace files contain frame distributions using a common GOP format, changing the fps value does not affect the statistical behavior of RCBR. However, it would indeed change the peak and average data rates of the traces from the values given in [ROS97]. We use our data rate and fps values consistently throughout this document without loss of any generality.

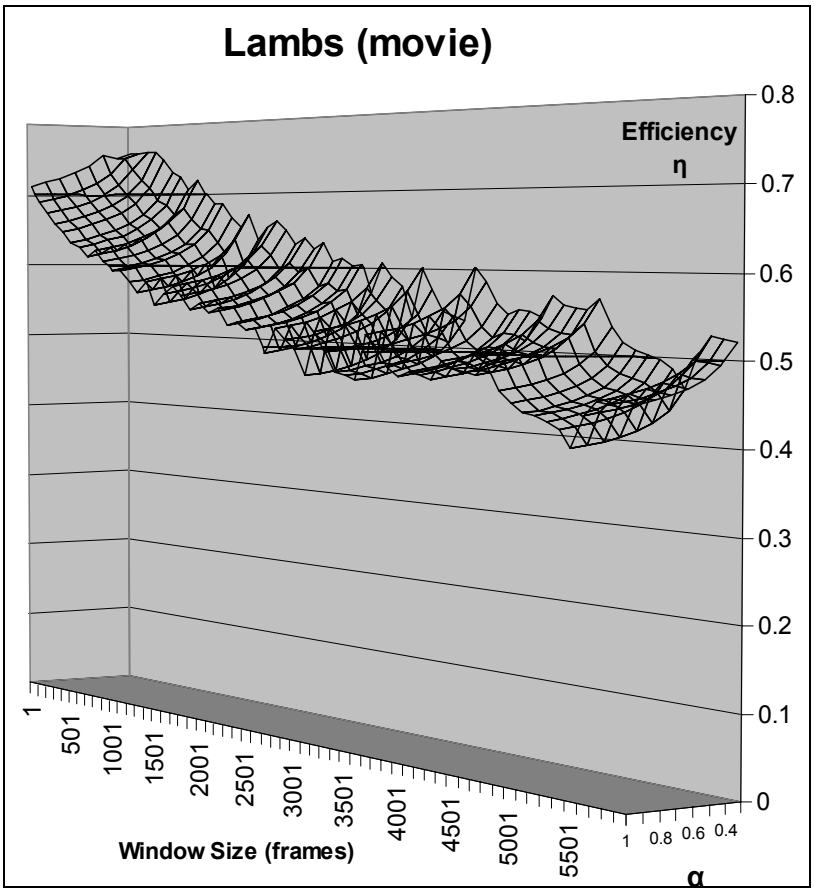
Also worth mentioning is amongst the movie traffic traces which we analyzed, there is little correlation between the film genre and burstiness. An intuitive guess always relates an action or sports genre movie with being very bursty and a talking head show such as a news or talk show as having a lower peak – average ratio. This intuition is not correct as such a correlation has not been noticed looking at the traffic traces. For example, '*Asterix*' (cartoon show) and '*Talk-show-1*' proved to be in the highly bursty category, contrary to their genre classification, whereas action movies such as '*Lambs* (*Silence of the Lambs – a thriller genre*)' and '*Bond* (*James Bond action movie*)' proved to be amongst the less bursty movies and thus showed a high degree of bandwidth savings in contrast with their genre type.

The next set of figures show a more normalized measure of determining RCBR's effectiveness by plotting the efficiency 'η' against window size (in frames) and the generalization parameter 'α'. The efficiency is simply a parameter that reflects what fraction of ideal savings we obtain by choosing a particular window size and α value. Ideal bandwidth savings would be obtained when window size equals one i.e. the bandwidth is renegotiated after sending each frame. The general trend observed is as expected i.e., efficiency of using RCBR decreases as α is lowered. This is due to lesser bandwidth reserved with a lower α. Plots for lower, moderate and high burstiness are shown in figure 3.5 below.


Mr. Bean (TV Comedy)
Efficiency
η
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
600
1100
1600
2100
2600
3100
3600
4100
4600
5100
5600
Window Size (frames)
1
0.8
0.6
0.4
α


Simpsons (cartoon show)
Efficiency
η
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
101
501
901
1301
1701
2101
2501
2901
3301
3701
4101
4501
4901
5301
5701
Window Size (frames)
1
0.8
0.6
0.4
α


Bond (Action movie)
Efficiency
η
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Window Size (frames)
100
600
1100
1600
2100
2600
3100
3600
4100
4600
5100
5600
1
0.8
0.6
0.4
α


Lambs (movie)
Efficiency
η
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Window Size (frames)
1
501
1001
1501
2001
2501
3001
3501
4001
4501
5001
5501
1
0.8
0.6
0.4
α

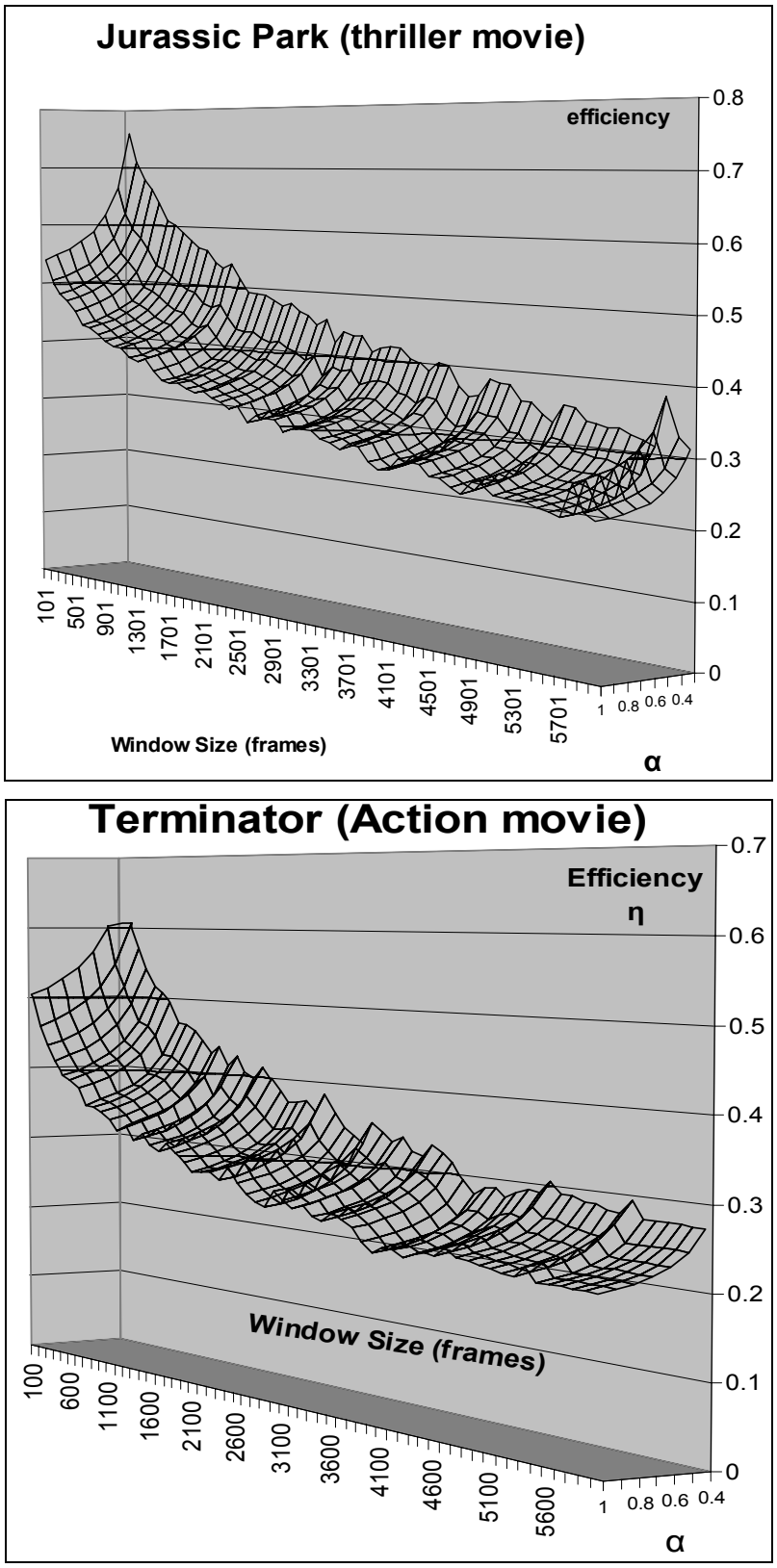


Figure 3.5: Efficiency of videos with calm, moderate and intense scene activity

3.4 Discussion of results

Through our analysis we have shown that RCBR is very effective in optimizing bandwidth usage on the network. Along with being easy to implement for streaming pre-recorded video, its performance improves with more flows due to the statistical multiplexing gain as will be shown in chapter 5. The extra bandwidth released by flows becomes available for the others. It has the capability to work with existing QoS protocols such as RSVP and MPLS. We have specifically worked with RSVP in this study but it could easily be implemented to set traffic reservations for various classes defined for Differentiated Services. RCBR can work on top of any buffering mechanisms used to smooth the VBR stream. Although we presented the results for all practical values of window sizes, we recommend using RCBR with a time-window equal in size to a multiple of the QoS refresh period (typically 30 seconds for RSVP). This would ensure no extra overhead while updating bandwidth reservations.

RCBR concept is well accepted but it has not been implemented on a real test-bed to study its performance. Commercial VoD servers (such as Kasenna®) prefer to use application level protocols such as RTP for maintaining flow integrity and assume sufficient backbone bandwidth for requested clients. These servers are used mostly on dedicated (IP based) cable TV networks on different data channels so there is little need for using QoS mechanisms. Video traffic characteristics are not considered at all so there is a strong need to create a database of traffic trace files that may be used by intelligent applications to implement RCBR technology to maximize their network utilization for the same installed capacity. The preliminary simulation results obtained motivated us to create a small network test bed with multiple senders and receivers and observe the actual network response to using RCBR on a loaded network. The experimental setup and logistics are discussed in the next chapter.

CHAPTER 4 Experimental Model and Implementation

Initial experimentation with RCBR generated results indicating that up to 60% of the reserved bandwidth can be saved while viewing less bursty videos such as 'Mr. Bean' and 'Simpsons' etcetera. This chapter discusses the need for implementing the RCBR mechanism on a real network. There is also a discussion of our experimental test bed setup, logistics, software implementation and approximations.

4.1 Need for experimentation

The RCBR relies on the temporal and bursty nature of pre recorded video and has been successful tested only through simulations [MSD97, KT97]. This mechanism heavily relies on the availability of a traffic pattern database that contains traffic information about compressed video files. Such a type of bandwidth renegotiation system was not tested neither implemented in actuality because

(1) RSVP is not widely deployed on public routers and is disabled by default.
(2) Reservation refresh messages were considered a means of exchanging control information rather than bandwidth updates.
(3) Renegotiation intervals were not universally decided upon by any standard and so the computation intelligence could not be embedded in the servers
(4) Renegotiating bandwidth at arbitrary and varying intervals was looked upon as added extra control overhead especially for a per-flow handling scheme such as RSVP.
(5) Behavior of the network was not defined for RED (Reservation Establishment Delay) and RTD (Reservation Teardown Delay) in relation to number of flows [MM01]. Bandwidth changes were looked upon as adding extra delays to the flows.
(6) Streaming video servers do not extensively use bandwidth reservation technology, because most of the networks they operate on are dedicated cable networks with excess capacity.

(7) Streaming video servers lay more emphasis on MPEG decoding efficiency, buffering and application level reliability protocols such as RTP than QoS protocols.

Here are some arguments against the points mentioned above. Though RSVP is not supported widely on public routers there is good potential of it being used on privately owned networks such as universities, corporate campuses and neighborhood Internet networks which are managed centrally and follow common configuration policies. In such a scenario, increased development of RSVP based applications would prove fruitful. For RSVP, the periodic refresh messages are considered as a control mechanism but no extra overhead would be caused if the periodicity of bandwidth updates were a multiple of the natural refresh interval for RSVP. In the case of DiffServ QoS implementations where network traffic flows are bound to classes each assigned a quantitative reservation level, the video traffic may safely be demoted or promoted in class to match the bandwidth needs. Instead of directly fetching data from the network buffer if the applications are made to fetch data periodically from an intermediate buffer then the problem of RED (Reservation Establishment Delay – typically around 100ms) would be solved. The sending applications would adjust their sending rate after taking the RED into account. Except the case with download-and-play mechanism (where in reality the video streams from the local disk) no buffering mechanism for a streaming video can result in a constant bit rate data stream. Even with the case of PCBR [CK96] the variation in bandwidth requirements over various sections of a video stream would enable a group of flows to perform well with the statistical multiplexing gain introduced by RCBR. Thus RCBR would work on top of any currently used buffering or smoothing mechanism.

Popular industry standard streaming video servers such as Kasenna® rely on application level mechanisms such as RTP (Real Time Protocol), RTCP (Real Time Control Protocol) and RTSP (Real Time Streaming Protocol) to transfer data streams over a connectionless (UDP) or connection oriented protocol (TCP). These also concentrate more on MPEG decoding efficiency and efficient buffering mechanisms rather than QoS provision. These features ensure a timely in-order packet delivery and error correction.

Mostly these servers are used on private and dedicated VoD networks which are often unshared so QoS is not a significant need. For example, at the Alexandria Research Institute of Virginia Tech, cross campus video conferencing is done through a dedicated ATM line. Since the video stream is independent of shared traffic there is no need to use QoS because performance does not suffer from bandwidth contention. Any gains obtained by efficient buffering and decoding could be complemented by RCBR because it scales over any magnitude of bandwidth variation. If the future of VoD over IP networks has to be fortified then services using shared networks must be optimized. Examples of these services would be on-demand online courses available over a campus network, recorded conferences or sessions over a private company network and the popular video on demand for IP based cable operators. Implementation of QoS especially using RCBR is absent in commercial implementations of VoD services so our research work aims at testing RCBR over a real test-bed network and assess any performance benefits obtained.

4.2 Test bed architecture

Chief demonstration aims for the experiment were to implement RCBR successfully over a real network and compare the performance of the system in terms of packets lost/delayed while using and not using RCBR mechanism for a number of parallel video streams. The network in our test bed aimed at emulating a group of nodes on one end of the network receiving data from a VoD server at the other end of the network and separated by at least three router hops. A simplified but generic diagram depicting the network is shown in Figure 4.1.

Besides the physical connectivity of the network shown in Figure 4.1, implementation of all the components shown were software based. Ease of obtaining drivers and installing QoS modules for a Microsoft Windows operating system made it a very strong choice for us. We did experiment with the ISI's KOM-RSVP engine to be installed on the UNIX based routers to make them QoS aware but hardware incompatibility of our 802.1p network cards with Unix prevented us from going forward with Linux based devices.

Moreover the API support for RSVP and SBM available in Microsoft was far better than the RAPI module (RSVP API) available for UNIX. All the application development for Server and Client modules was also done on Visual C++ platforms. All the factors combined made Microsoft Windows as our chosen operating system for the test bed

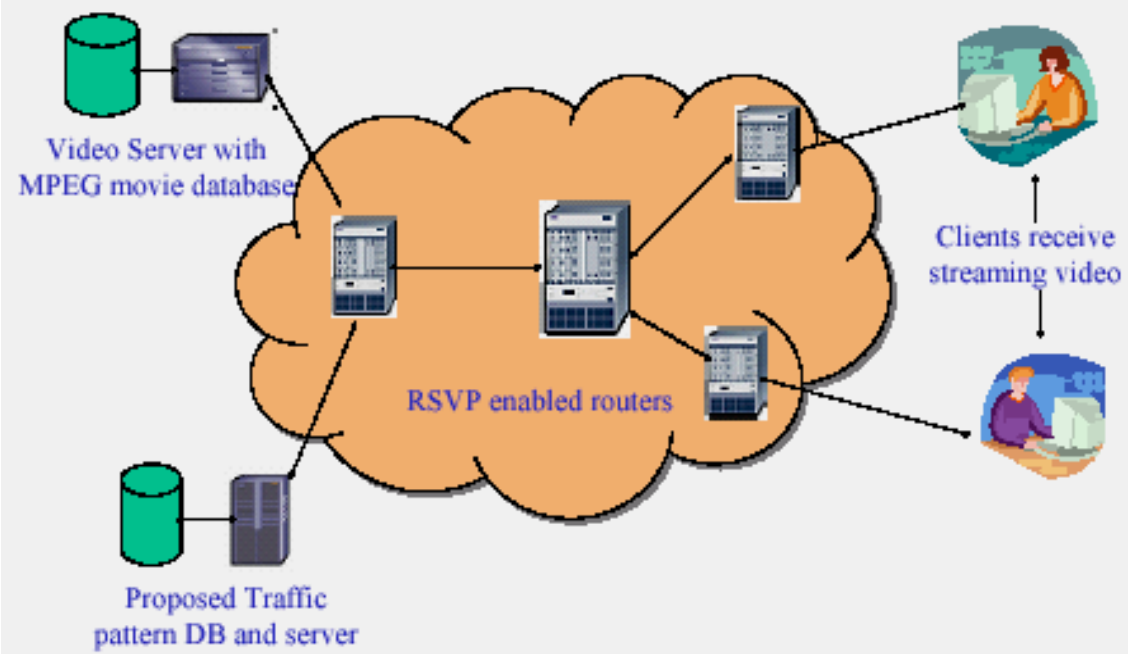


Figure 4.1: Emulated test bed for implementing a VoD service with RCBR

.The primary protocol that works in congruence with the RSVP protocol is the SBM (Subnet Bandwidth Manager). Its operation is described in detail in chapter 2 and RFC 2814. SBM makes it possible to use RSVP across multiple subnets connected by layer 2 (switches) or layer 3 (routers) devices. Each subnet has at least one SBM aware device chosen as the Designated SBM (DSBM) that is responsible for admitting or rejecting bandwidth requests on its network segment as well as maintaining the reservation states of individual flows. The DSBM makes its reservation decision using preset policies that contain parameters such as total and individual bandwidth rates (in bytes/sec), burst size allowed (in bytes) and peak bandwidth occupied (in bytes/sec). The policies can be specified using the SBM module interface (also called the QoS Admission Control) as shown below in Figure 4.2. When a DSBM is present on a (sub)network it becomes a 'managed subnet.'

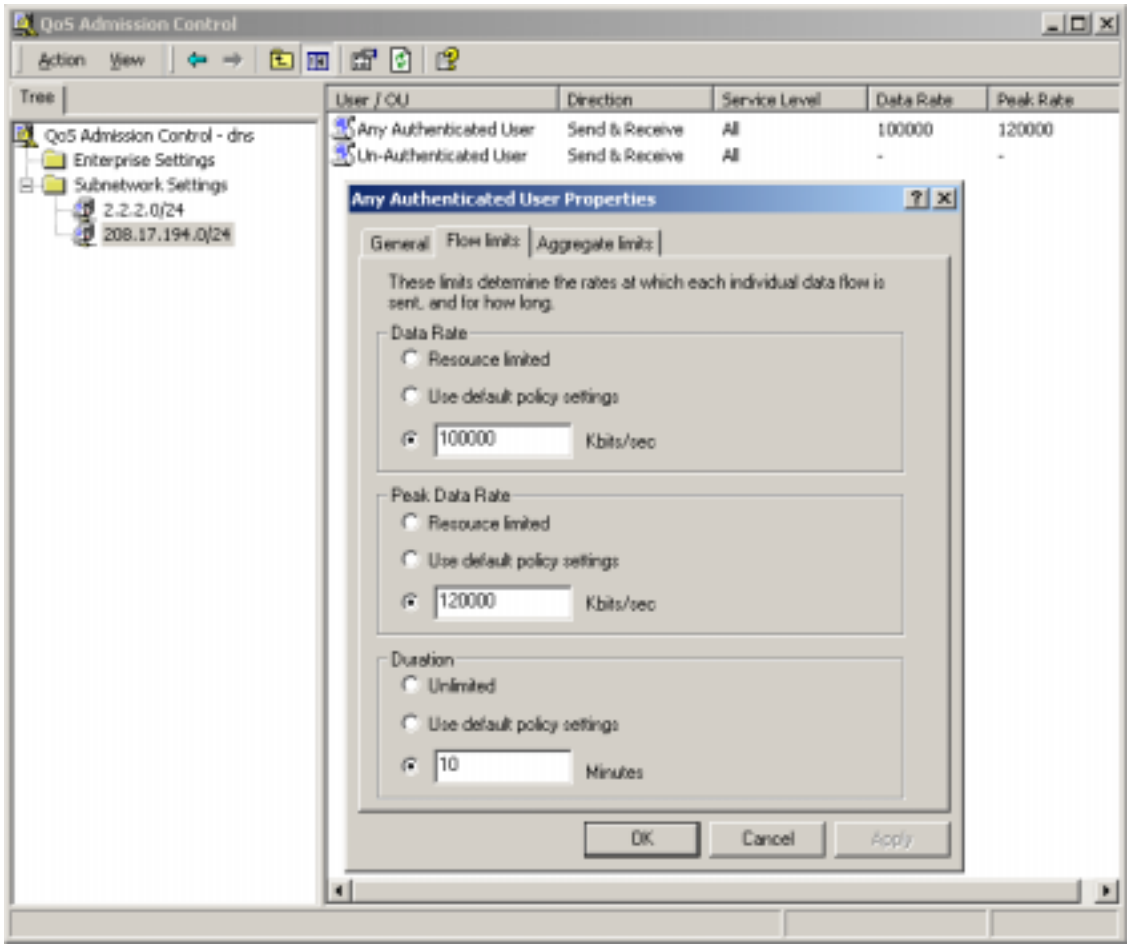


Figure 4.2: Control module for setting SBM parameters

It becomes easier to manage settings and user groups especially when the nodes on the managed network are included in the same 'domain.' The domain system is for organizing nodes in a large network in a hierarchical and orderly fashion so that it becomes easy to manage them, name them and access them using their qualified host names. This is referred to as the 'Active Directory' system. This system can be managed in a better way by installing a domain name server on the subnet and giving an arbitrary domain name to the network hosts. This involves a lot of steps while configuring the hosts in a managed subnet. The DSBM needs to be setup as a DNS server and Active Directory (AD) needs to be installed on it so that different hosts in the network can be made a part of the AD. For each host an account need to be created on the DSBM. This may be done using another GUI panel as shown in Figure 4.3 below.

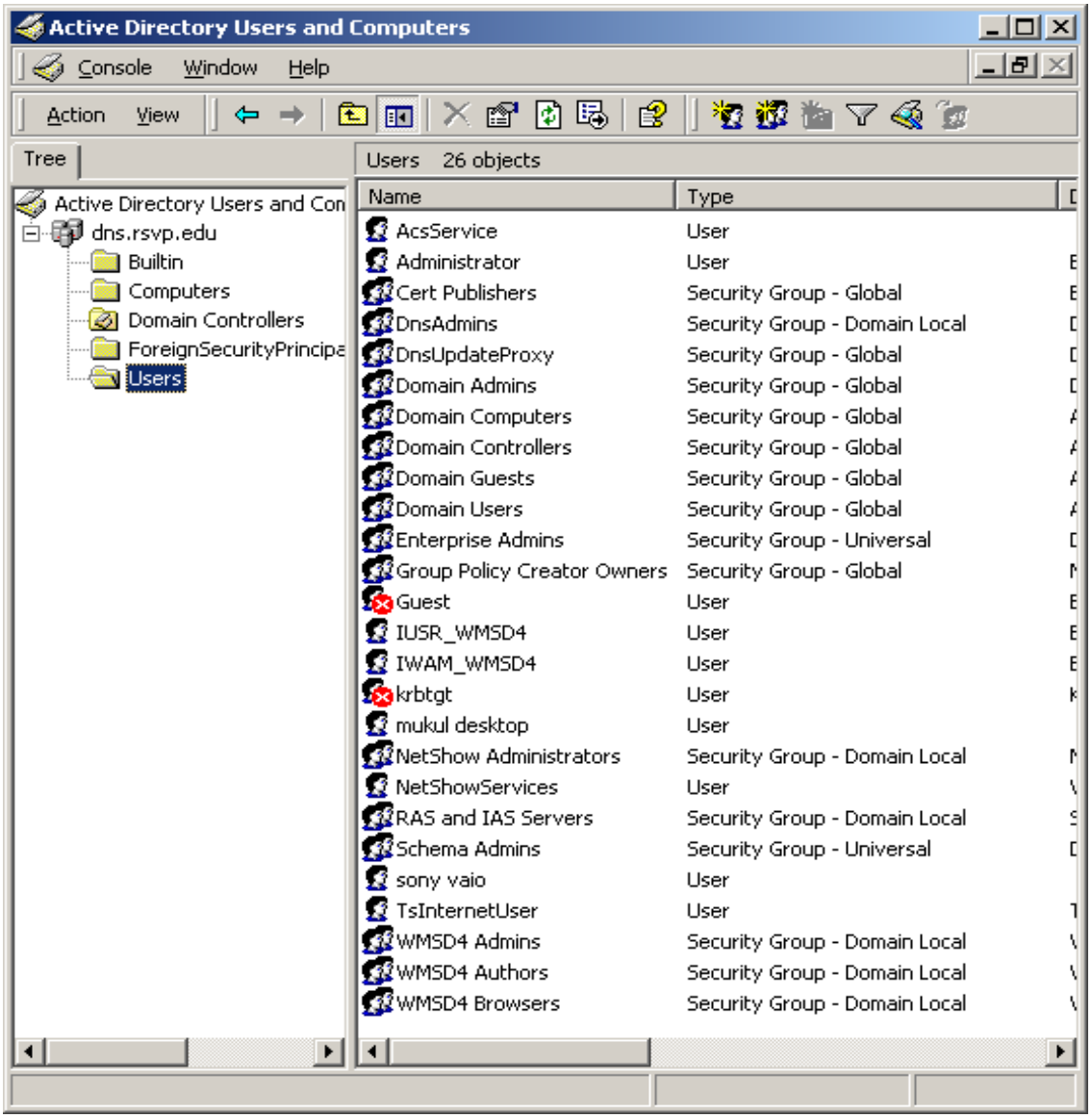


Figure 4.3: Control Panel for configuring and adding hosts to the Active Directory

In our test-bed the hosts had static IP addresses and they needed to be mapped with their host names. That was done using the DNS control panel interface as shown in Figure 4.4. The host names seen (VAIO, 2000-IS-BEST and WMSD4) were the machines used with their names statically mapped with their IP addresses so a DNS lookup on them would be possible.

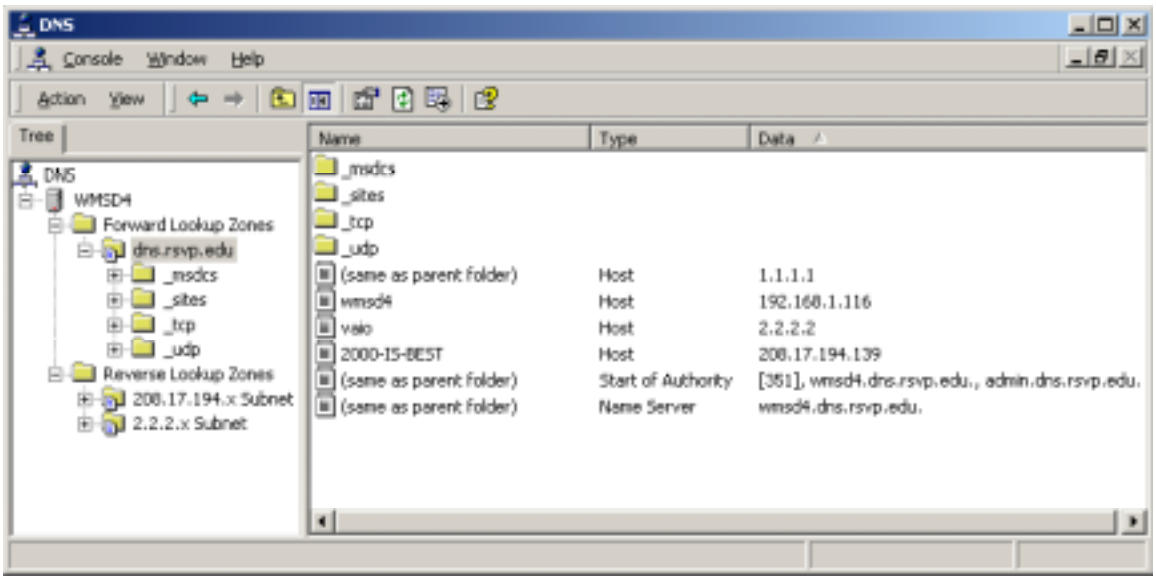


Figure 4.4: Domain Name System (DNS) control panel

Finally the QoS RSVP service had to be configured on all hosts and the DSBM. For this the QoS packet scheduler module needed to be installed for the network driver. The QoS packet scheduler is responsible for flow control, queue management and packet shaping while sending packets of a data stream at a preconfigured flow rate. The RSVP service provider (RSVP-SP) is a process in Microsoft Windows that is responsible for implementing the RSVP protocol according to RFC 2205. All application level QoS calls have to be made to the RSVP-SP, which in turn interfaces with the network link layer. This service is automatically invoked when any QoS specific call is made by an application. It may be manually started from the Services Control Panel as shown in Figure 4.5. An important configuration of RSVP-SP is to run it as the 'AcsService' user account, which stands for the admission control service user account.

Implementation of test-bed and assumptions

The network described in Figure 4.1 was simplified for test purposes without causing any lack of accuracy in results. The video server was implemented as a multithreaded application on one host and the receivers were implemented as multiple processes on another client machine. All clients being on the same host did not make any difference to the operation of RSVP or SBM because each individual flow is characterized by the IP

address – port combination of sender and receiver. Since all receivers were operating on different UDP ports their bandwidth reservation requests were treated as separate ones by the intermediate DSBM.

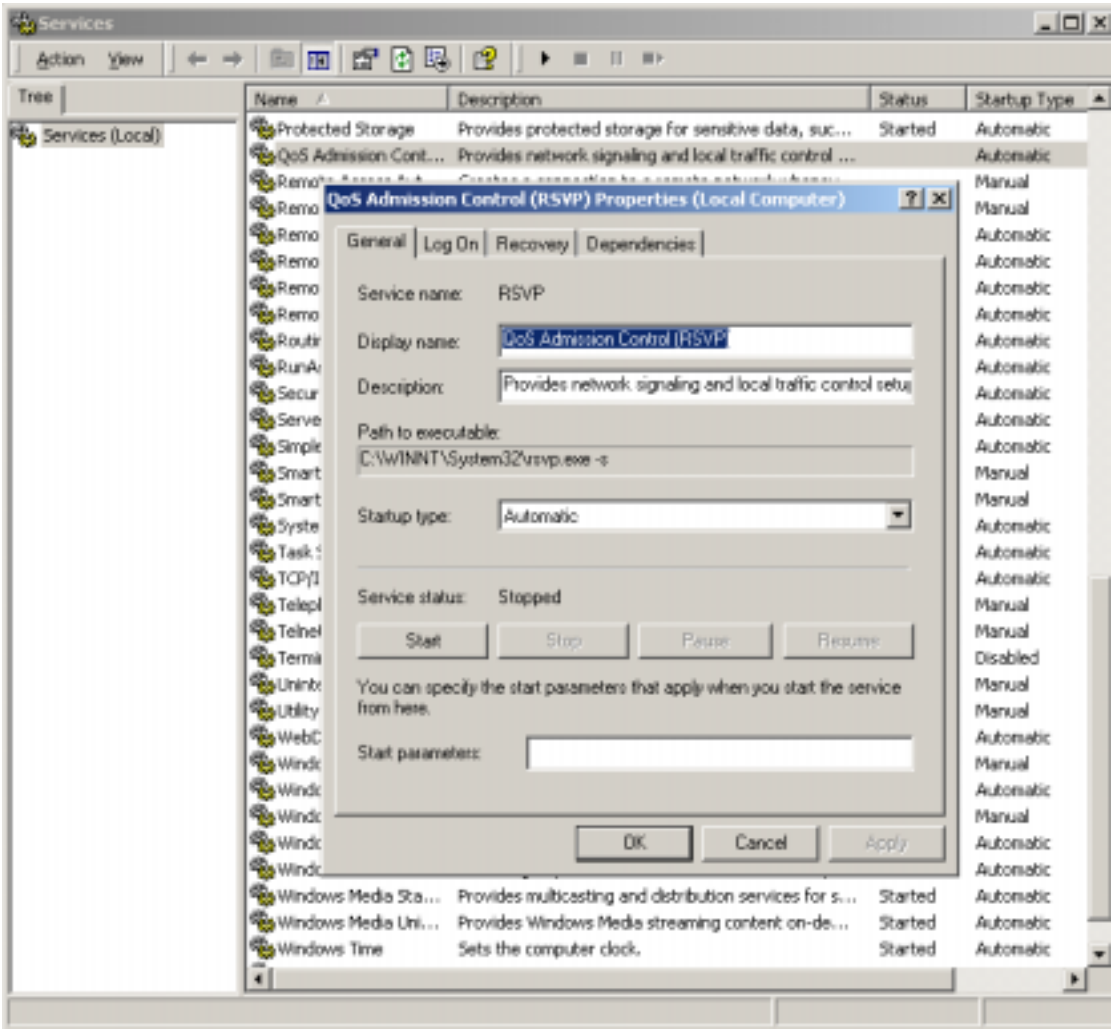


Figure 4.5: Control Panel showing QoS RSVP Admission Control Service menu

To demonstrate the behavior of a DSBM towards QoS RSVP requests, a single router hop was kept between sender and the receivers. This single router was a multifunctional node because it was the DNS domain controller, the DSBM, the router and the gateway between the server (sender) and clients (receivers). Keeping just one node as a single router hop and DSBM was kept not only because of simplicity of the network but also because of equipment limitations. Although the RSVP functionality worked perfectly, for

the SBM protocol to work we needed a non-DSBM gateway router each for both server and clients. This is because the gateway edge router has to send the PATH message to the receiver's IP address after doing a routing table look-up instead of sending it to the AllSBMAddress (224.0.0.17). Since the DSBM coincides with the edge router, it forwards the sender's PATH message to the reserved AllSBMAddress and consequently the receiver is unable to detect the PATH message intended for itself. Figure 4.6 shows the real setup that we used for the experiments.

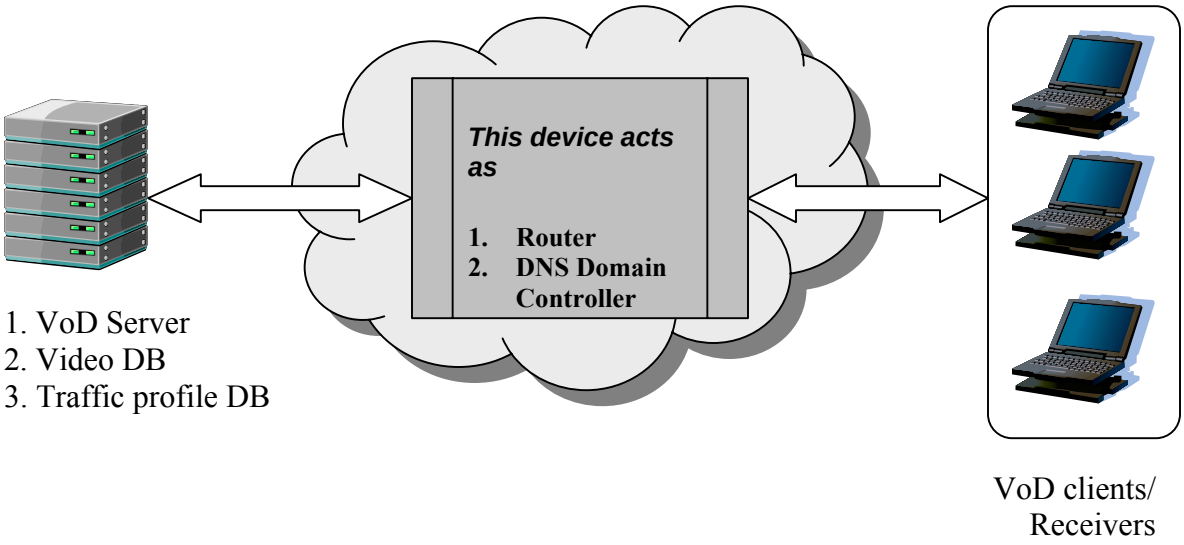


Figure 4.6: Our implementation of network shown in Figure 4.1

To solve this problem a suitable work around way adapted was to write our own Subnet Manager with the same functionality as a DSBM. This module could be located anywhere on the subnet so that any host could contact it requesting bandwidth approval before issuing a relevant RSVP flowspec. In this way we could propose added functionality to the DSBM besides just allowing or rejecting QoS requests. A detailed discussion on the additional features added is mentioned in the next sections.

4.3 Algorithms and software implementation

Since we focused purely on the networking aspect of VoD we did not consider most of the application layer programming such as GUIs, MPEG decoding and screen playback.

The applications were win32 console applications that displayed network statistics such as number of bytes sent, token rate reserved and error information, if any. The server is written as a multithreaded application that spawns a new thread for each client's video request. It confirms the requested video's availability, looks up its traffic file to obtain the necessary parameters (token rate) in order to send out QoS requests and then begins sending the data stream. Since we aim at comparing performance of CBR versus RCBR we had to make two types of senders – Static Sender (that used the CBR type of bandwidth reservation) and Dynamic Sender (that used the RCBR scheme). The receiver application remained the same in both cases. Our applications used the APIs (Application Programming Interface) provided by the Microsoft implementation of QoS called the GQoS API (Generic Quality of Service). This API contains useful calls for making a network application QoS aware so that it may initiate and respond to QoS signaling.

Since RSVP is a receiver oriented protocol (i.e. the reservation level is actually requested by the receiver and not the sender) it was not an obvious choice to write one common receiver and two senders. The explanation is that both the PATH and RESV messages contain the FLOWSPEC object that contains both the sending and receiving 'flowspecs.' The sender's sending flowspec (in the PATH message) contains the QoS parameters recommended by the sender to the receiver so the latter has the choice to either accept these parameters as its receiving flowspec (in the RESV message) or overwrite them with its own. The same applies to a sender's receiving flowspec and receiver's sending flowspec. In our case, since the receiver does not have any of its own flowspec preference, the QOS-SP copies the sender's flowspec data into the receiver's flowspec data and eventually these are the values that are used to request reservation over the network.

4.3.1 Algorithm for RCBR implementation

RCBR implementation would mean reading one time window worth of data from the video file at a time and sending it after updating the current QoS guarantees on the network. Some important implementation issues that crop up are application level flow

control, buffer size and level of QoS for each window. These issues are dealt with in the following subtopics.

4.3.1.1 Buffering tradeoffs

Our RCBR implementation uses fixed time windows (equal to 30 seconds of video) during which the bandwidth reservation remains constant. As discussed in chapter 3, this constant bandwidth level is chosen assuming all frames are α $(0<\alpha \leq 1)$ times the size of the largest frame in the window. It is worth mentioning that the lowest α can go is dictated by the average frame size within the window. Using the lowest value of α would effectively mean that we are using a buffer equal in size to the bytes contained in the entire window (say 30 seconds of playback time). This would imply draining that buffer at a constant average rate either on the network or the client playback device. This buffer may be as large as 18 MB for videos at high resolution with sound, which makes it unfeasible for IP-TVs or IP based digital cable receivers with limited memory. Thus there is a significant tradeoff between the data stream smoothing (i.e. less burstiness) achieved by buffering and the latency and memory space required by the buffer.

4.3.1.2 Smoothing of data streams

Typically data bytes written to a network by a sending application are immediately sent on the network by the underlying protocol stack. If we use TCP, this could lead to congestion control mechanism being triggered off causing delays thus making TCP an unsuitable transport level protocol to use. On the other hand if we use UDP then large packet bursts could lead to large packet losses, which would make the playback quality suffer. Even under QoS guarantees with RSVP the network can only take a finite burst size and support a limited peak data rate (as per the token bucket model explained in chapter 2). Any further burstiness would cause packets lost (UDP) or delayed (TCP). Thus there is a strong need to smoothen the data flow over the network while using minimal buffer requirements. Two popular methods of doing so are (1) implementing smoothing at the network layer, and (2) implementing smoothing at the application layer.

4.3.1.3 Network Layer Smoothing

Packet stream smoothing at the network level is done using the packet scheduler and packet shaper modules of the underlying TCP/IP stack. In Microsoft implementations the packet shaper/scheduler modules are easily installed using Network Connection Properties panel as shown in figure 4.7 below

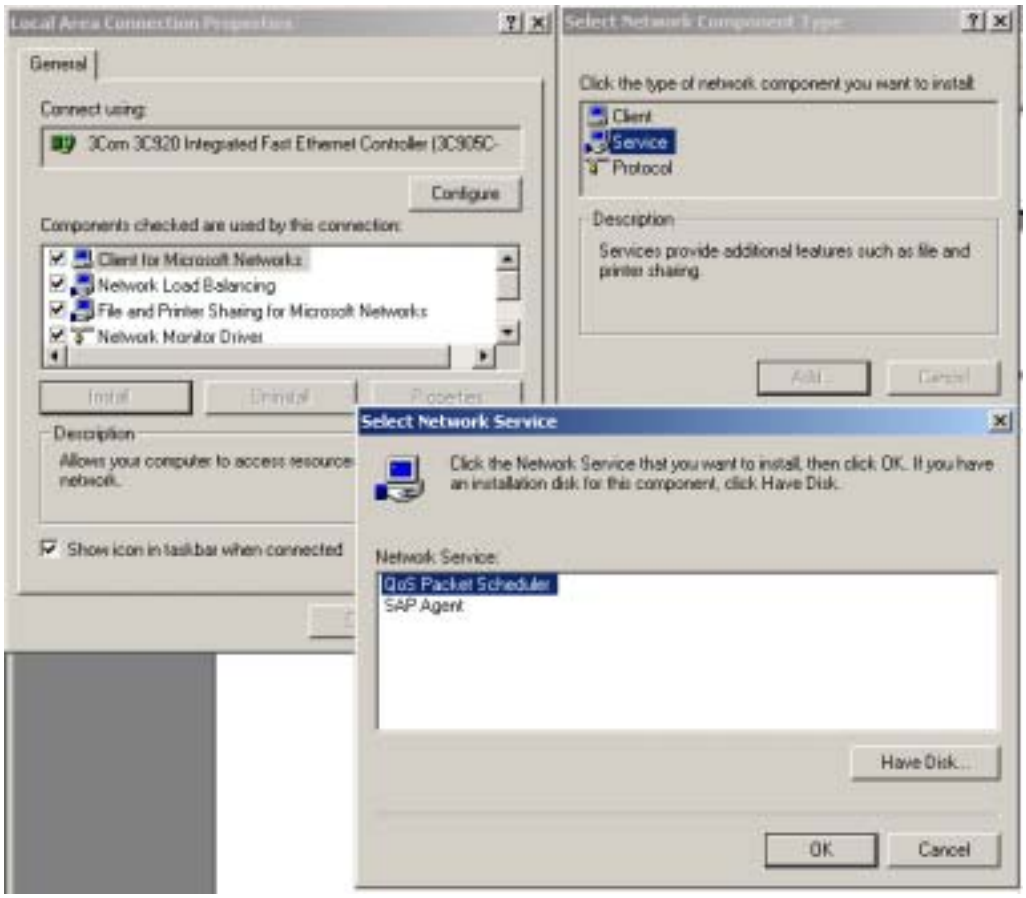


Figure 4.7: Installing the Packet Scheduler

The network packet scheduler is invoked whenever an application specifies the 'Shape Discard' object in the provider specific buffer of the QOS data structure. The QOS data structure is an important part of the GQOS API that holds important flowspec information as well as special objects specific to the QoS service provider (which in this case is the Microsoft QOS-SP). The packet scheduler may be configured giving any of the following options:

Table 4.1: Various configuration options of the packet scheduler module

TC_NONCONF_BORROW	Instructs the Packet Shaper to "borrow" remaining available resources *after* all higher priority flows have been serviced.
TC_NONCONF_SHAPE	Instructs the Packet Shaper to retain packets until network resources are available to the flow in sufficient quantity to make such packets conforming. (For example, a 100K packet will be retained in the Packet Shaper until 100K worth of credit is accrued for the flow, allowing the packet to be transmitted as conforming).
TC_NONCONF_DISCARD	Instructs the Packet Shaper to discard all nonconforming packets.

The packet shaper's job is to break down large packets into smaller ones so that they conform to the specified network load. The packet scheduler's job is to send out the shaped packet from its queue in accordance with the network constraints. In our case we used the **TC_NONCONF_SHAPE** mode to avoid losses with the TC_NONCONF_DISCARD mode and bandwidth surges rising above a window's CBR level while using the TC_NONCONF_BORROW mode. A VoD server's packet scheduler and packet shaper modules work by buffering and queuing packets from all sending threads internally before shaping them to a smaller size and releasing them onto the network according to a specified time schedule to maintain a specified flow rate. The performance of a server can suffer significantly if the number of packets accumulated in its shaper reaches high numbers. The packet scheduler overload is typically caused when applications release all their data in a burst to the network stack and expect the packet shaper to solely regulate network flow according to the CBR level specified. In our case the sender application reads data contained in a window and sends it immediately over

the network. If there is no application level flow control the packet scheduler can become overwhelmed with data packets and may lead to either longer delays or packet bursts/losses. Snapshots of TCMON packet scheduler are shown in figure 4.8(a)-(b) that indicate the amount of packets accumulated in the packet shaper with 16 simultaneous flows using application level buffering (fig 4.8(a)) and without using application level buffering (Fig. 4.8(b)).

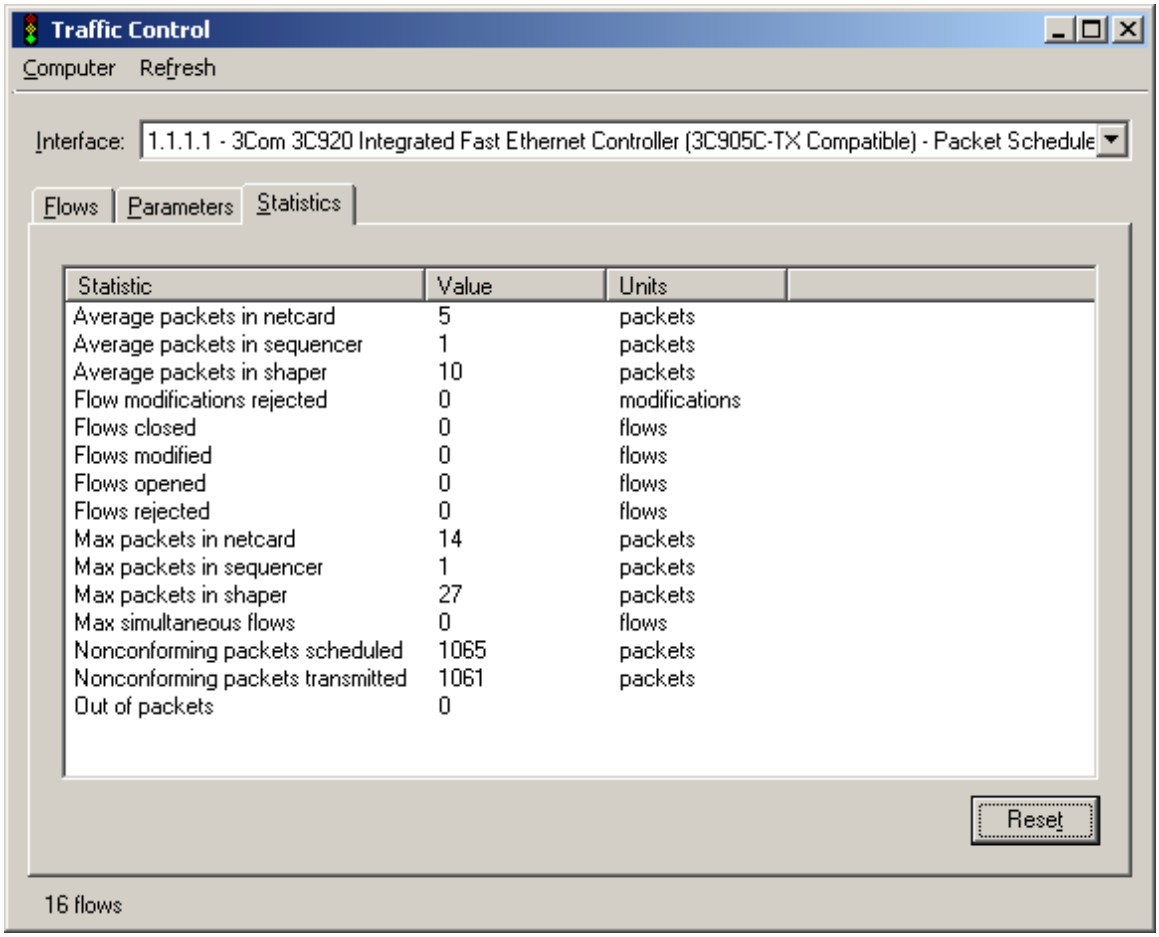


Figure 4.8(a): TCMON (Traffic Monitor) sniffer with application level buffering

This snapshot was taken after about 5 minutes of the flows starting. The statistics to note are the 'Average packets in shaper' and 'Nonconforming packets scheduled.' Compare these values with the ones shown in Figure 4.8(b).

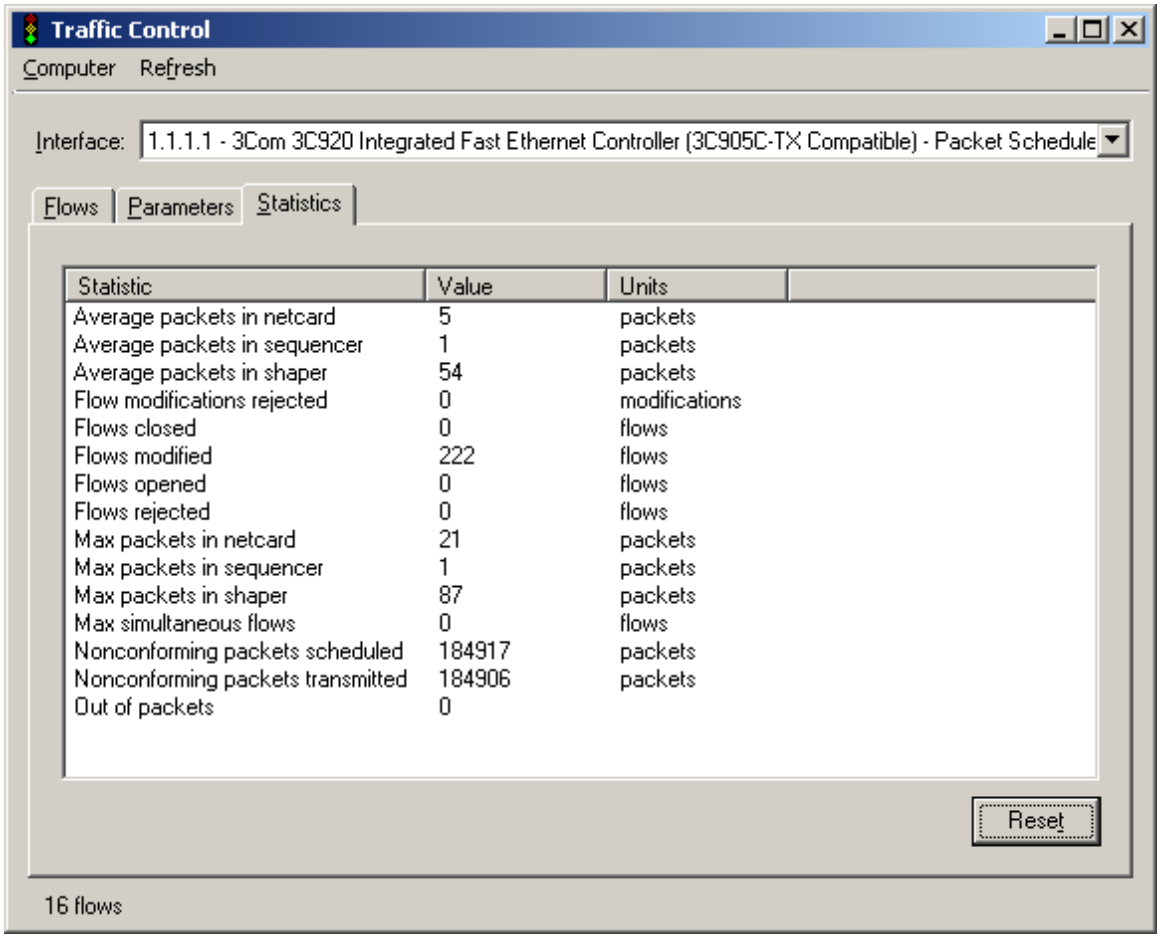


Figure 4.8(b): TCMON (Traffic Monitor) sniffer without application level buffering

4.3.1.4 Application Layer Smoothing and Concept of Smoothing Windows

Network level smoothing would get complemented if the sending application exercises some flow control mechanism while writing data bytes to the network. Buffering is an unavoidable means of smoothing a data stream. While sending a window of data maximum efficiency is obtained when the flow rate equals the average data rate. Since smoothing the entire RCBR window (that carries 30 seconds of video) was ruled out due to buffer and latency issues we considered smoothing even smaller windows within the RCBR window as our smoothing unit. We call them **Smoothing Windows (SW)** and their size (in frames) can be chosen so that they are a factor of the RCBR window size (900 frames). Our sending application would compute the average frame size for each of these SWs and send the data contained within them in packets equal to their average frame size. Since the frame rate of MPEG encoded videos is constant at 30 frames/sec

each window (RCBR or smoothing) would have a designated time period for sending the frames contained in the video. After sending the MPEG frames for each SW our server would wait till the corresponding time period is over. This would enable the network packet scheduler to deal with limited amount of data at one time and need for shaping would be lesser as the packets would already be broken into smaller sized one by the application. For example if the smoothing window size is 5 seconds then the network packet scheduler would have to deal with only 5 seconds worth of data as compared to the whole 30 seconds worth of data in a RCBR window if no application level smoothing was done. The performance benefits can clearly be noticed as the SW size decreases and the average number of packets in the packet shaper decreases (as shown by the TCMON sniffer). In our implementation the minimum value of SW is 1 second of playback time that equals to 30 frames.

4.3.2 Choosing the RCBR Token Rate

According to our algorithm, for each 30-second RCBR window the RSVP token rate reserved is computed as the maximum of all Smoothing Window (SW) data rate averages. Let *'RCBR_BW'* be the token rate reserved for each RCBR window with window size = '*N*' smoothing windows. Let '*m*' ($30 \le m \le 900$) be the number of frames (of variable size F_m) in each SW, then equation 3 gives the reserved token rate for every RCBR window as the maximum of all average frame sizes of the N smoothing windows contained inside the RCBR window.

$$RCBW\_BW = MAX\left(\frac{1}{m}\sum_{1}^{m} F_m\right)_1^N \qquad \ldots(3)$$

Note that in our RCBR implementation we used a constant renegotiation interval of 30 seconds at 30 frames/sec rate so N = 900 MPEG frames. This is to enable the RSVP refresh messages to be used as means of updating bandwidth. In context of the plots shown in chapter 3 our operating point on the X-axis is window size (=N) of 900 frames.

4.3.3 Dynamic Smoothing Window Concept

Instead of keeping the Smoothing Window size constant, extra benefits may be achieved if the Smoothing Window (SW) size is kept flexible. The premise behind this choice is that as we increase the sample size of a collection of bursty data the statistical mean (first moment) of that data will 'most likely' be lower than that for a higher sample size. This may not be strictly true universally but considering the GOP (Group of Pictures) pattern used in MPEG encoding where most of B, P frames are much smaller than the I frames, adding more GOPs to our sample size would bring the average frame size closer to B or P frame sizes rather than I frame size. Applying this analogy to the smoothing windows concept, as we increase the size of a smoothing window the average frame size for that window would be lowered. This is shown in the plot below for the movie 'Asterix.'

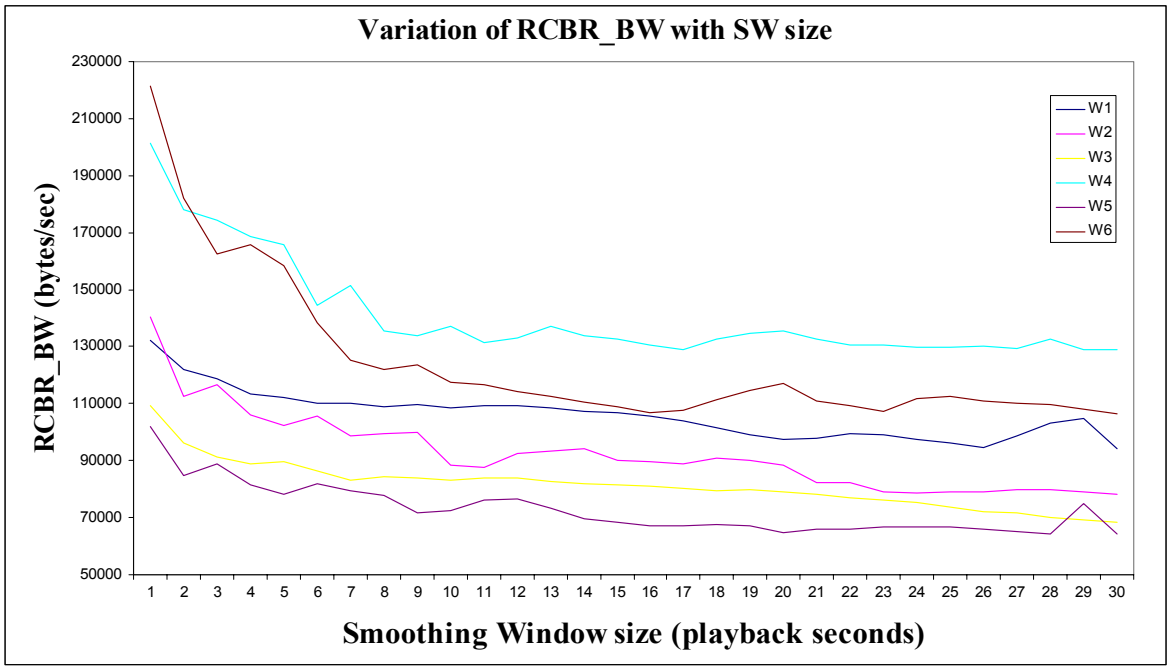


Figure 4.9: Effect of increasing the SW size on RCBR bandwidth level

The smoothing window size is actually the buffer that we use over the network and, as the plot in Figure 4.9 shows, a larger SW size leads to lower bandwidth requirements for the RCBR window. According to our implementation the minimum SW size is 1 second of playback time and the maximum is restricted by the size of the RCBR window (which in our case is a constant – 30 seconds of playback time). With SW = 1 second, application level smoothing is minimal. Also observed is that the smoothing effect

remains more or less unchanged beyond SW = 10 seconds. Thus we can safely fix an upper bound on our SW size as 10 playback seconds.

4.3.4 Adaptive Buffer Window Mechanism and Resource Tracker Agent

Using a buffer (or smoothing) window of 10 seconds for optimal application level smoothing would come with the tradeoff of a 10 second start-up delay for video playback, which we consider acceptable for most VoD situations. However as mentioned in section 4.4.1.3, writing large amounts of buffer data at once to the lower network layers (which in our case would be 10 playback seconds) would cause significant load on the packet shaper and scheduler. In the case of a video server serving hundreds of clients this could lead to memory bottlenecks, disk paging and possible latency in the video stream. Thus smoothing buffers cause scalability problems for the packet shaper module.

Here we propose a mechanism to mitigate this scalability tradeoff by using flexible buffer windows for every flow. We call it Adaptive Buffer Window (AWB) mechanism. We use the term 'buffer window' interchangeably with 'smoothing window.' The way this mechanism works is by setting the buffer window equal to an initial value, say 5 seconds, at the starting of each RCBR window and increase it later with the purpose of accommodating a flow that would otherwise be denied the requested QoS. As mentioned before, increasing the buffer window decreases bandwidth requirement for that RCBR window. With the current implementations of the Subnet Bandwidth Manager, flows requesting QoS are either granted or denied resources. There is no quantitative negotiation involved. To complement the Adaptive Window Mechanism we also propose a central Resource Tracker Agent (RTA) that could reside anywhere on the subnet to let QoS requesters know of the available bandwidth. This agent could also be implemented as a new object residing in the SBM protocol message.

As mentioned in section 4.3, test bed limitations made it impossible for us to integrate the SBM protocol with RSVP so we had to use our own resource allocation tracker in lieu of the DSBM. This RTA kept a table of current flows and their allocated QoS bandwidths (BW). The unique feature of the RTA is its ability to return the available-bandwidth

value to the senders as compared to the SBM's ability to simply grant or deny reservations. This is possible as the RTA records the total usable BW for QoS and the BW already allocated to current flows. Incase the available BW was lower than the requested BW then the ABW mechanism would kick in to find out whether increasing the buffer size would permit using the available BW for the flow. It would go on expanding its buffer window till the highest permissible value to match the available BW. If even its lowest BW does not fit then the flow would be forced to use its previous reservation level suffering packet losses. Any data sent in excess of the RCBR reservation level would be subject to either a best effort delivery by the network or discard depending on the option chosen from Table 4.1. The highest buffer window levels for different flows could be set to different values as they may be kept configurable on the client side.

It is important to note that since we keep a provision for expanding the buffer window to a maximum value later, the pre-fetch delay for a VoD client would be commensurate with that caused by the highest buffer level. This would be because the client's playback device would be configured to poll its application buffers every 'M' seconds, where M is the maximum buffer window level permissible. This would ensure no discontinuity in video playback, as the smoothing delay would always lie between know limits. The two advantages in using a smaller than maximum buffer window are (1) reduce load on the system packet shaper and packet schedule by introducing application level buffering, and (2) give the flows flexibility to increase their buffer size incase of BW contention so more flows may be accommodated for the same available bandwidth.

Like most streaming video mechanisms implemented, all frames arriving after their arrival batch are discarded. Lost frames are the parameter we present in the next chapter to compare the performance of our RCBR implementation versus keeping a static bandwidth level for the entire playback duration of the movie. We assume that a suitable application layer protocol exists to enable the VoD clients to communicate with the server and send information such as the maximum receive buffer and video title.

4.3.5 Smoothing Window File

The current traffic trace files used by the server exist as a single column of frame size values. Using these, a sender application needs to compute RCBR and buffer window levels on the fly during playback. With the ABW mechanism kicking in, these files would need to be extensively and redundantly referred to in order to compute QoS levels for different buffer windows. This could become another bottleneck in terms of scalability. We propose a new file format containing pre-calculated data ready to be used by the ABW mechanism. We call it the **Smoothing Window File** (**SWF**). This file contains a grid reflecting QoS guarantees (reserved bit rate) for every combination of a RCBR window and buffering window. Since we keep the RCBR window fixed at 900 frames we have 30 columns for each buffer window ranging from 30 frames (1 second playback time) to 900 frames (30 second playback time). The rows reflect currently playing RCBR windows, e.g. 1-900, 901-1800, 1801-2700 and so on. As compared to the approximately 250KB video traffic trace file obtained from [ROS97] the SWF file is less than 10KB in size. Every time the sender chooses a new buffer window it just looks up the QoS level for it instead of scanning through the 250KB traffic trace file. Also worth mentioning is that the plot in Figure 4.9 is made using the data from certain RCBR windows of a SWF. The SWF file is shown in Figure 4.10 where the file contents have been pasted into a spreadsheet for easy view.

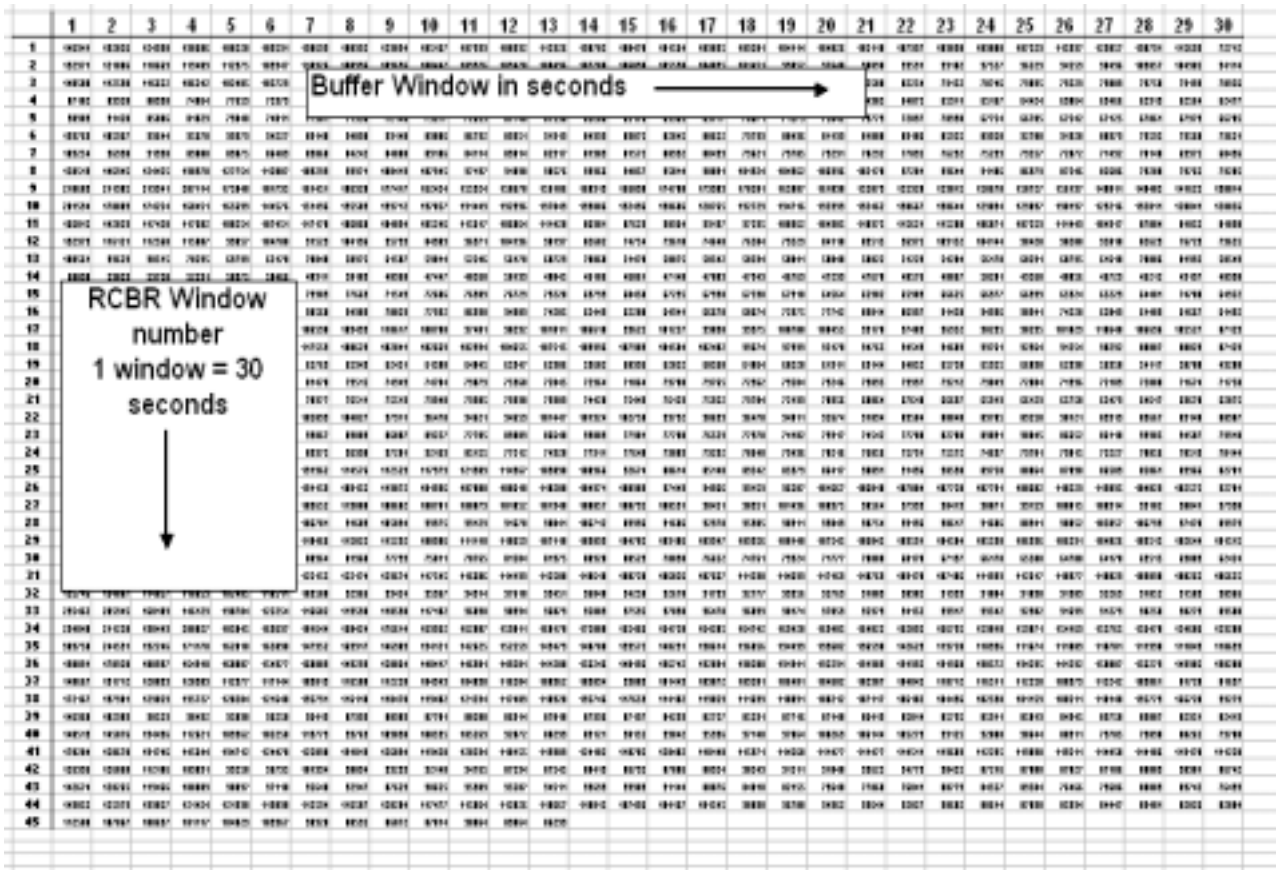


Figure 4. 9: Snapshot of the entire SWF file pasted in a spreadsheet

4.4 Flowchart depicting logic implementation by the VoD server

A flowchart showing the sequence of events that take place while sending the data file over to the client is presented in Figure 4.11. It shows various stages of the sender. Some parts worth noting are:

(1) Resetting of the buffer window back to its initial value (5 playback seconds or 150 frames) when a new RCBR window is beginning to be sent

(2) Sequential increment of the buffer window when available BW as shown by the RTA is less than requested. Incase the buffer window reaches its maximum value (10 playback seconds or 300 frames) and the available bandwidth is still insufficient then the sender maintains its previous QoS level at the maximum window size before beginning to send the data.

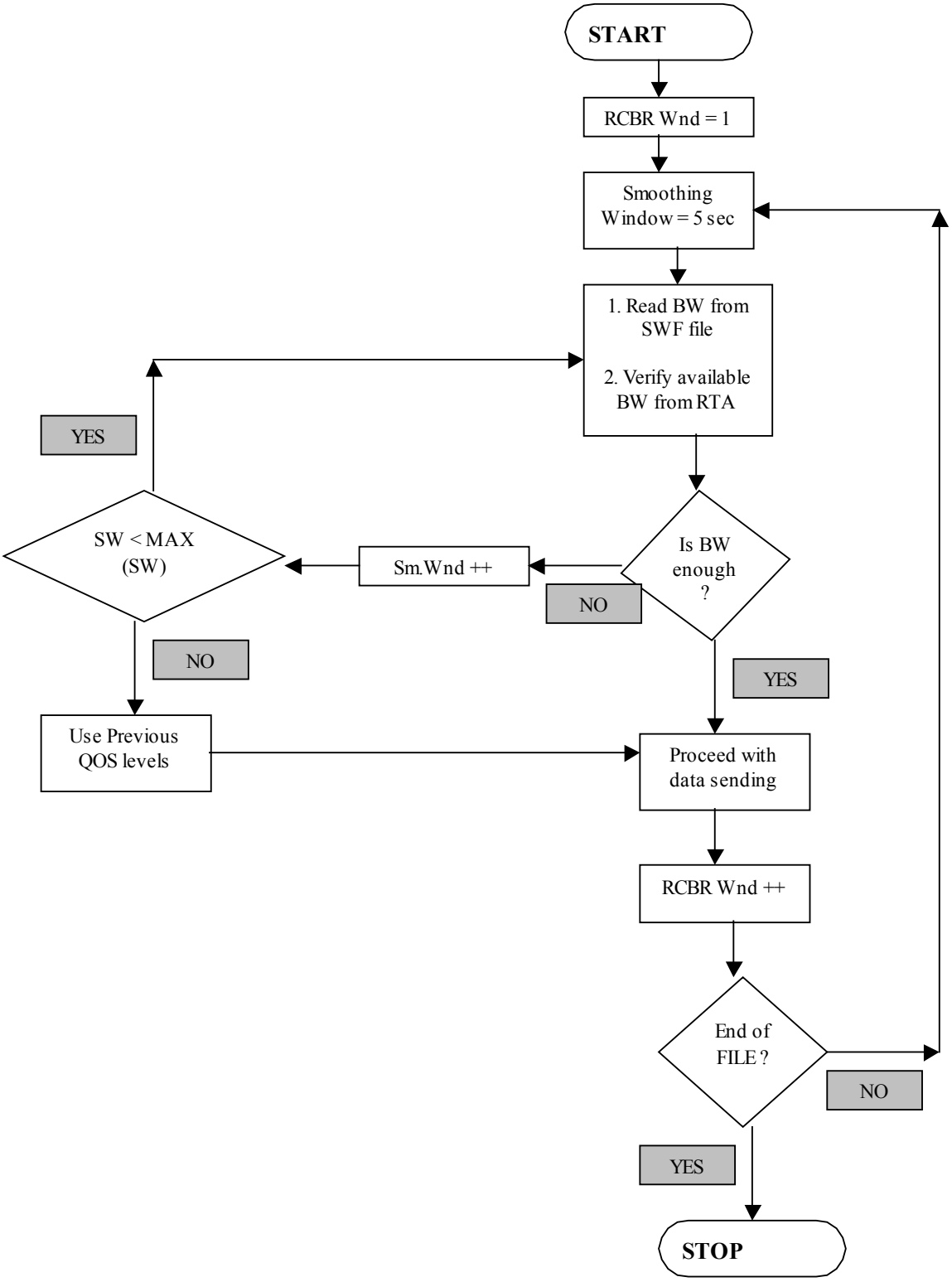


Figure 4. 10: Flowchart depicting logic implementation by the VoD server

4.5 Summary

The flowchart above depicts the working mechanism of our RCBR implementation. We have proposed some new features to improve the RCBR implementation on a Video-on-Demand server. These include:

1. Application level buffering using smoothing windows;
2. Adaptive Buffer Window (ABW) mechanism to give flexibility for accommodating flows that would otherwise have been denied reservation;
3. Resource Tracking Agent (RTA) to act as a substitute of SBM on a subnet and return a quantitative value of available bandwidth to the sender application; and
4. Smoothing Window File (SWF) format for giving sender applications quick access to pre-computed QoS level data while adjusting the sending buffer window

The next chapter contains the results obtained by our implementation which show that for the same amount of reserved QoS bandwidth RCBR flows give lower packet losses than the conventionally used static QoS level applications.

CHAPTER 5 – Experimental Results and Analysis

In this chapter we discuss our experimental approach towards gathering data using our test bed. We describe the traffic trace files used for the videos, the parameters used and test cases considered while collecting empirical results for comparing CBR based bandwidth allocation and our implementation of RCBR.

5.1 Description of Traffic Trace Files

The MPEG encoded files have a predictable data flow pattern, as it is relatively easy to extract the encoded frame information out of them. Since they have a fixed playback rate in terms of frames per second, knowledge about the individual frame sizes directly leads to the data rate that needs to be sent over a network while transferring these files. We obtained some frame traces of MPEG-1 encoded single layered videos from [ROS97] which were generated using the UC Berkley MPEG-1 software encoder [KLG94]. The generated traffic files were encoded using the following parameter set:

Group of Pictures (GOP) Pattern: IBBPBBPBBPBB (12 frames)
Quantizer scales: 10 (I), 14 (P), 18 (B)
Motion vector search: logarithmic/simple
Window: half pel
Reference frame: original
Encoder input: 388x288 pels with 12 bit color information
Number of frames per sequence: 40,000 (about 22 minutes of video at 30fps)

The frame trace files contain a sequence of frame sizes in a human readable form. As an example here is pasted a list of frames from one random GOP as copied from the movie '*Starwars*': (the numbers show the frame size in bits)

…

…

…

34216 – I

3616 – B

3576 – B

7080 – P

3232 – B

2840 – B

7824 – P

2976 – B

2704 – B

7048 – P

2608 – B

2240 – B

…

…

…

Plots showing frame size distribution of these videos (such as the one shown in Figure 3.1) give us an insight into the burstiness of the movie. Some frame trace plots show smooth size distribution and oscillate around a constant level for long stretches of time whereas some frame traces show rapid changes in frame sizes of each type of frame. We usually relate the latter category of frame traces to sports and action movies such as 'Soccer' and 'Terminator.' However this is just an intuitive assumption as burstiness is introduced when the correlation between adjacent frames is low. This happens because the prediction images are poor and additional data has to be encoded to correct these prediction errors. Videos of calm genres such as art movies and talking heads can be very bursty attributing due to factors such as frequent scene changes and rapid camera movements. The characteristics of 'Talk-1' and 'Talk-2,' which are TV talk shows,

exemplify such phenomenon. We have experimented on the videos mentioned above and the results are presented in the following sections.

The characteristics of selected videos that we have used for experimenting on our test bed such as compression rate, mean bit rate and peak rate/mean rate are listed in Table 5.1:

Table 5. 1: Characteristics of MPEG encoded movies used in experiments

Sequence	Compression Rate X:1	Frames		Overall Bit Rate	
		Mean [bits]	Peak/Mean	Mean [Mbps]	Peak [Mbps]
Asterix	119	22348	6.6	0.670	4.421
Race	86	30749	6.6	0.922	6.072
Soccer	106	25110	7.6	0.753	5.709
Starwars	130	15599	11.9	0.279	3.744
TalkShow-1	183	14537	7.3	0.436	3.203
TalkShow-2	148	17914	7.4	0.537	3.983
Terminator	243	10904	7.3	0.327	2.387
	From [ROS97]			From frame traces	

5.2 Bandwidth constraint on flows

For both CBR and RCBR cases we used the same bandwidth constraint and used packet loss as a metric for measuring relative performance. Packet loss occurs when the bytes to send in a particular window of time exceeds what the bandwidth constraint allows. The sender application uses the permissible sending bandwidth and drops the remaining packets that do not fit in that window. To choose a suitable constant reservation rate for the CBR case we could either (1) choose the peak data rate for the movie consistent with the largest frame size, so the chosen rate would cause no packet loss even if all frames were equal to the largest frame, or (2) use the smoothing window mechanism to logically divide the entire movie into small smoothing windows and then choose the largest average data rate of all smoothing windows as the designated token rate. Choice (2) can be used without loss of generality because choice (1) would imply a zero buffer size on the server side, which rarely is the case. There always exists some level of application and network level buffering present on the VoD server side. In our case we choose a 5 second buffer consistent with our RCBR implementation's default buffer value as discussed in chapter 4.

Let BW_{CBR} denote the CBR bandwidth obtained by choosing the largest average data rate of each smoothing window. If the frames in each smoothing window were sent at their average rate, no packet losses would occur at BW_{CBR}. However since we aim to compare the performance of RCBR versus CBR in terms of packet loss under bandwidth constraints we reduce this level to a value $\mathbf{C*BW_{CBR}}$ ($0 < C \leq 1$) with the purpose of putting a constraint on the flow. We call C as the 'cap' put on BW_{CBR} and express it either as a fraction or as a percentage. Note that the purpose of this cap is identical to the parameter 'α' introduced in section 3.3. Since this cap is put on each flow individually the performance of any single flow is independent of other flows at any time. Packet loss results whenever the data rate exceeds $\mathbf{BW_{CBR}}$. There is no statistical multiplexing because we choose the TC_NONCONF_DISCARD option from Table 4.1 while implementing this scenario. Thus for N identical flows we have $\mathbf{N*C*\ BW_{CBR}}$ as the total reserved bandwidth and this value is used as the total available bandwidth for the

same N flows sent using the RCBR mechanism. If the N flows are different then **N*C* BW_{CBR}** should be replaced by Σ (Ci * **BWi_{CBR}**), where the 'i' subscript denotes the values for the i^{th} flow.

Thus the following parameters are kept constant in the CBR and RCBR cases while sending data from the server to multiple clients:

1. Total bandwidth resources available;
2. Number of flows;
3. Inter-flow startup delay;
4. Type of flows (videos);
5. Network infrastructure and background traffic;

5.3 Performance Comparison Data and Plots

For streaming media we consider data loss in the vicinity of 10% significant enough to cause a visible loss in viewable quality. As the cap (for CBR flows) is decreased the per-flow reservation (= C*BW_{CBR}) decreases, which causes further packet losses. We use the accumulated reservation level (**N*C* BW_{CBR}**) for our RCBR implementation and notice the packet losses to compare with the CBR case.

Here are the flows we considered the plots of which are attached on the following pages:

(i) 'Starwars' with 5, 10, 15, 20 flows and cap = 50%
(ii) 'Soccer' with 5, 10, 15, 20 flows and cap = 50%
(iii) 'Terminator' with 5, 10, 15, 20 flows and cap = 65% and 60%
(iv) 'TalkShow-1' with 5, 10, 15, 20 flows and cap = 50%
(v) 'TalkShow-2' with 5, 10, 15, 20 flows and cap = 65%
(vi) 'Asterix' with 5, 10, 15, 20 flows and cap = 50%
(vii) 'Race' with 5, 10, 15, 20 flows and cap = 50%

(viii) All of above flows present 1, 2, 3, 4 times and cap = 50%

The following statistics are noted for the experiments:

1. Number of flows
2. Bandwidth reserved via RSVP and SBM
3. Packet loss while using CBR mechanism and bandwidth mentioned in (2)
4. Packet loss while using RCBR mechanism and bandwidth mentioned in (2)
5. Percentage packet loss (average) while using CBR mechanism
6. Percentage packet loss (average) while using RCBR mechanism

We send these flows simultaneously but start each new flow at a multiple of our chosen RCBR renegotiation interval in order to maximize statistical multiplexing gain by aligning the bandwidth renegotiation boundaries of all flows.

(i) **For a medium bursty movie ('Starwars') the performance comparison is shown below:**

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	313180	18347060	723866	8.15%	0.330
10	626360	36694120	267128	8.15%	0.0610
15	939540	55041180	197300	8.15%	0.0299
20	1252720	73388240	48967	8.15%	0.0056

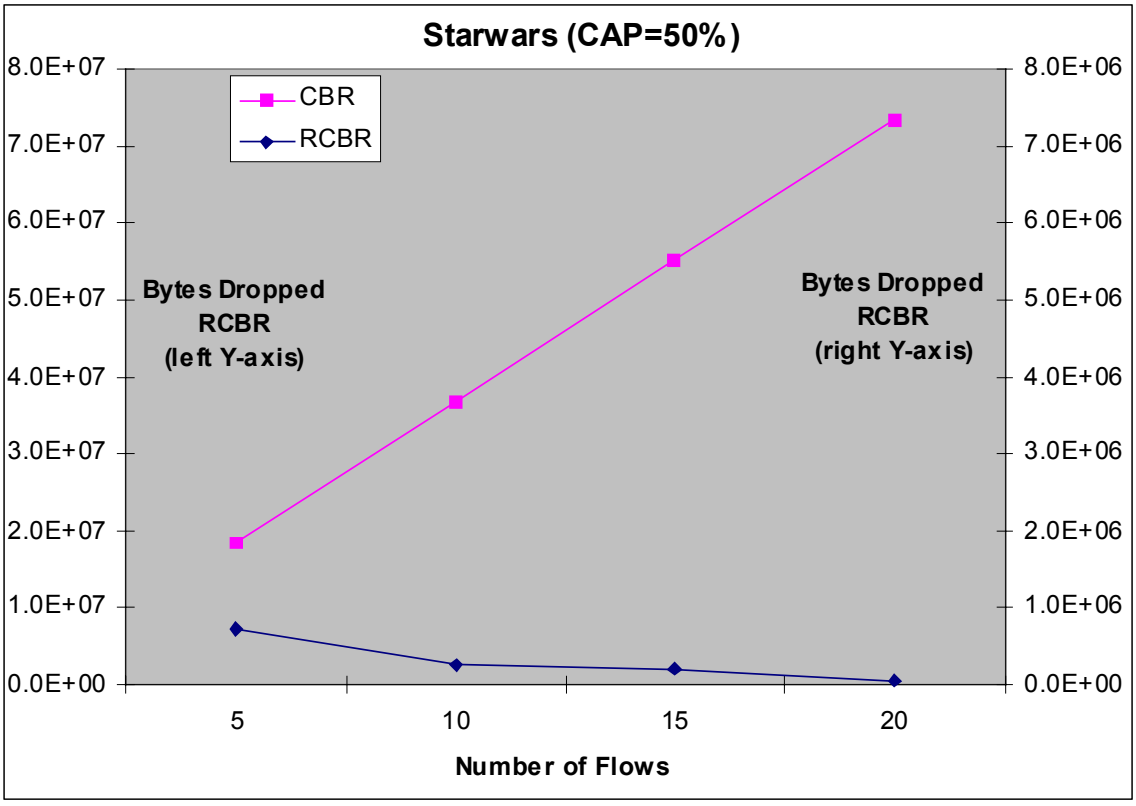


Figure 5.1: Comparison of CBR and RCBR mechanisms for 'Starwars'
Note the presence of two Y-axis scales, left for CBR and right for RCBR

(ii) **Statistics and plot for the medium bursty video 'Soccer'**

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	635605	23621135	18292730	3.96%	2.999
10	1271210	47242270	22690000	3.96%	1.859
15	1906815	70863405	20036155	3.96%	1.095
20	2542420	94484540	1833944	3.96%	0.0752

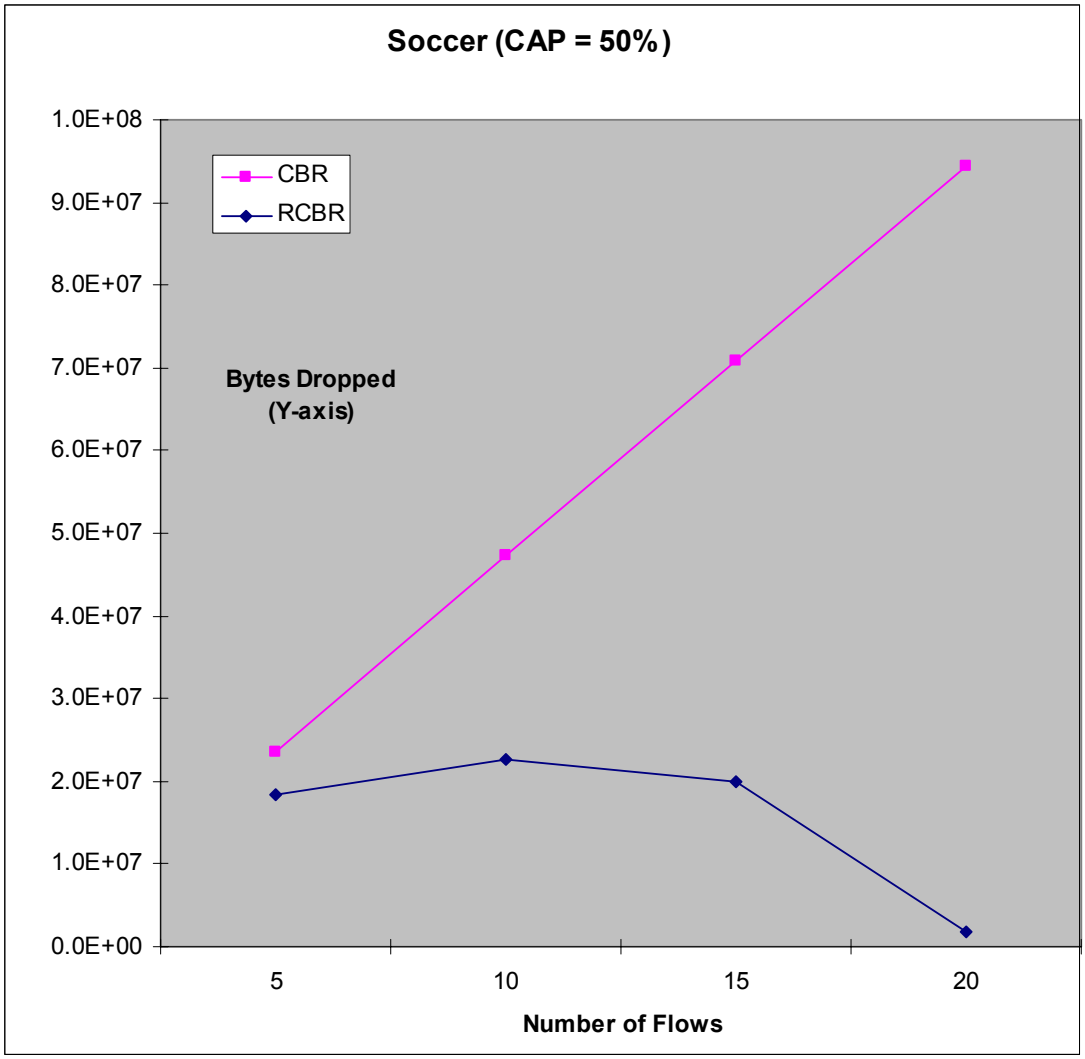


Figure 5.2: Comparison of CBR and RCBR mechanisms for 'Soccer'

(iii) **Statistics and plot for the high bursty video 'Terminator'**

(a) For cap = 65%

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	217160	17871985	339478	6.90%	0.1259
10	434320	35743970	445325	6.90%	0.0826
15	651480	53615955	736273	6.90%	0.0911
20	868640	71487940	236488	6.90%	0.0219

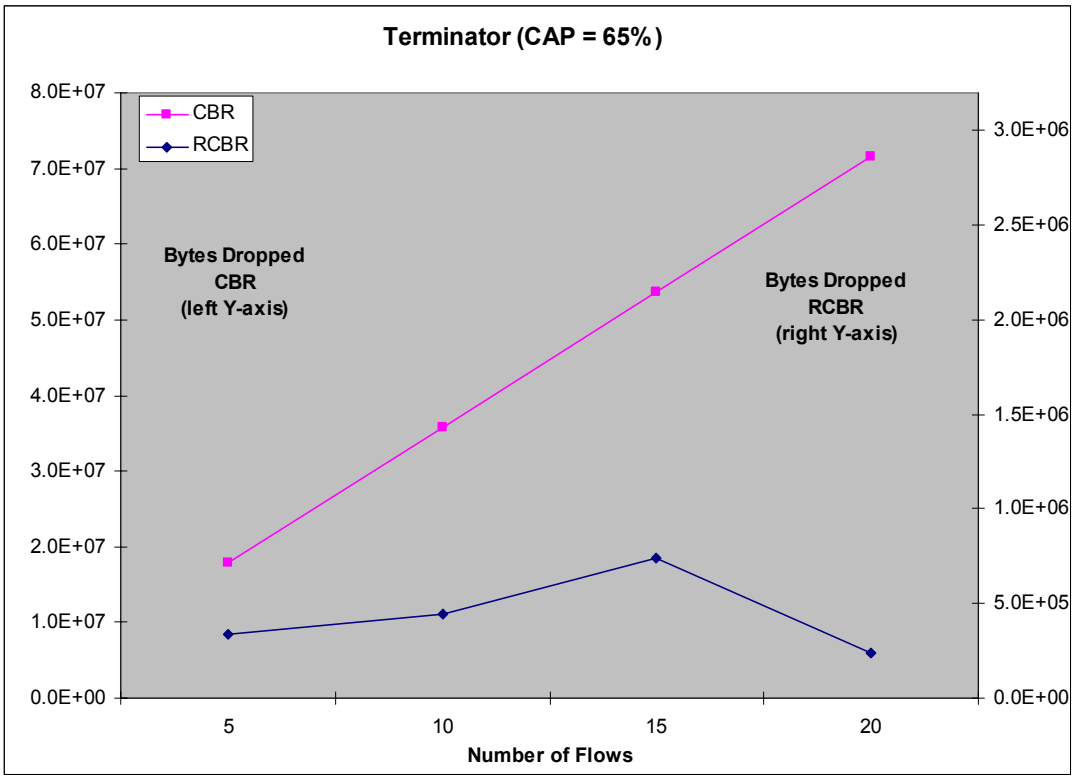


Figure 5.3: Comparison of CBR and RCBR mechanisms for 'Terminator' – 65%
Note the presence of two Y-axis scales, left for CBR and right for RCBR

(b) Plot of 'Terminator' for cap = 60%

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	200455	27980560	723866	10.77%	0.2686
10	400910	55961120	267128	10.77%	0.0496
15	601365	83941680	197300	10.77%	0.0245
20	801820	111922240	444319	10.77%	0.0413

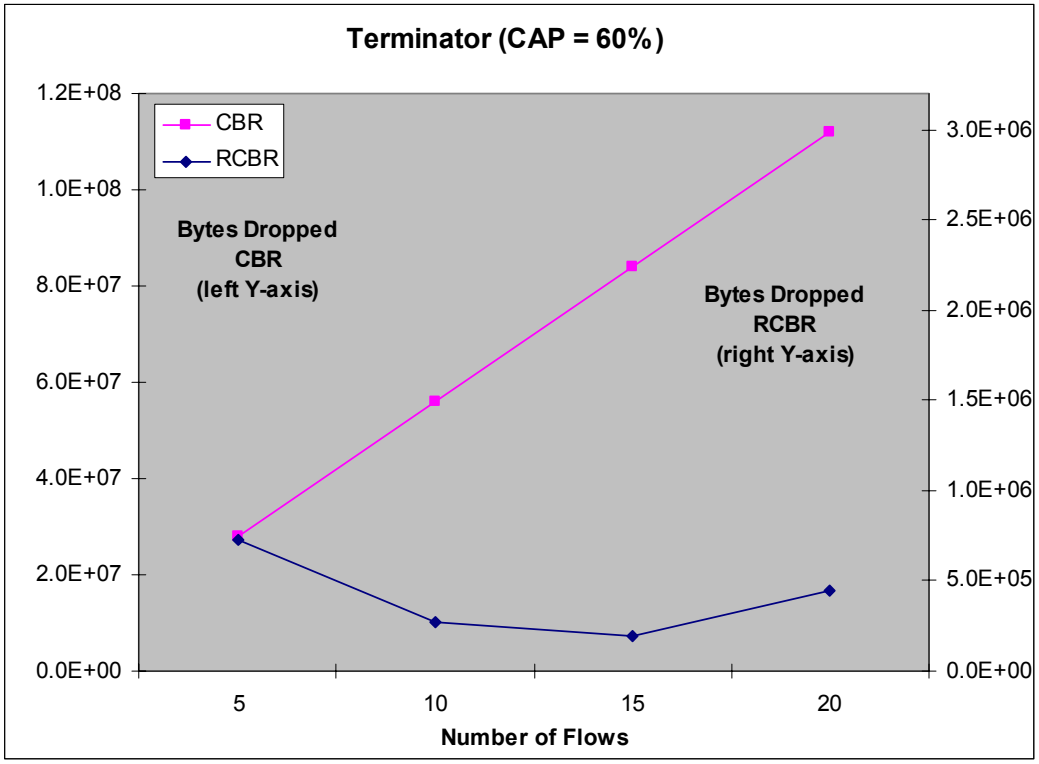


Figure 5.4: Comparison of CBR and RCBR mechanisms for 'Terminator' – 60%
Note the presence of two Y-axis scales, left for CBR and right for RCBR

(iv) **Statistics and plot for the high bursty video 'Talkshow-1'**

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	290150	29016215	1292730	8.35%	0.365
10	580300	58032430	1392142	8.35%	0.196
15	870450	87048645	1036155	8.35%	0.0974
20	1160600	116064860	595448	8.35%	0.0419

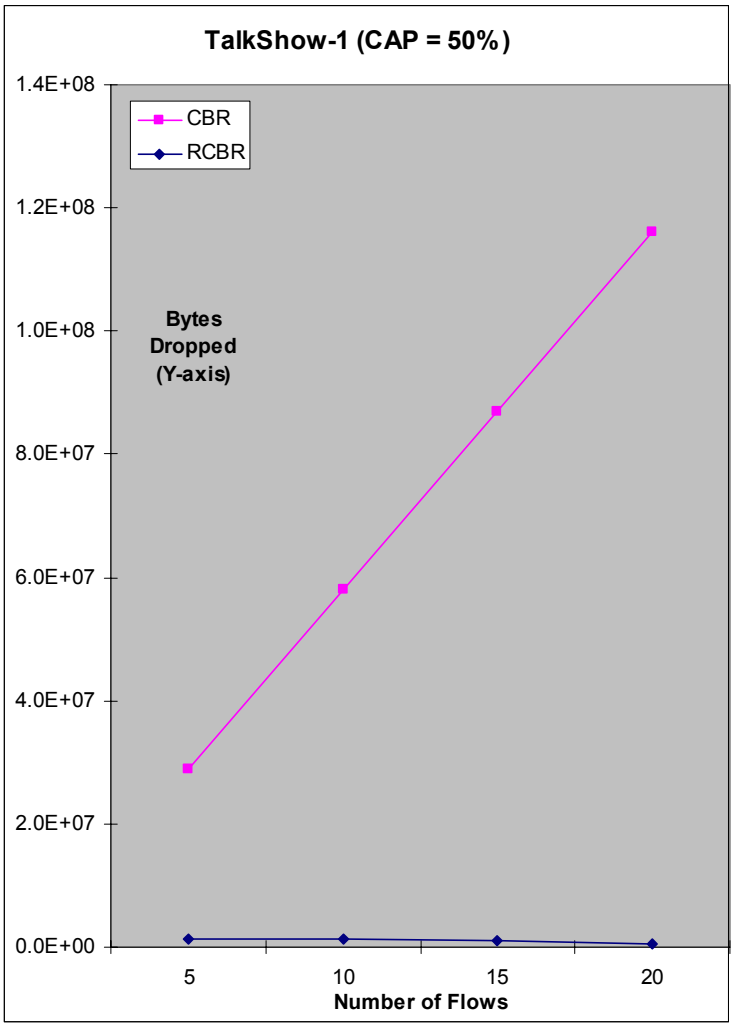


Figure 5.5: Comparison of CBR and RCBR mechanisms for 'TalkShow-1'

(v) **Statistics and plot for the high bursty video 'Talkshow-2'**

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	608440	2452960	2238591	4.62%	4.070
10	1216880	4905920	676720	4.62%	0.615
15	1825320	7358880	2000723	4.62%	1.213
20	2433760	9811840	795768	4.62%	0.362

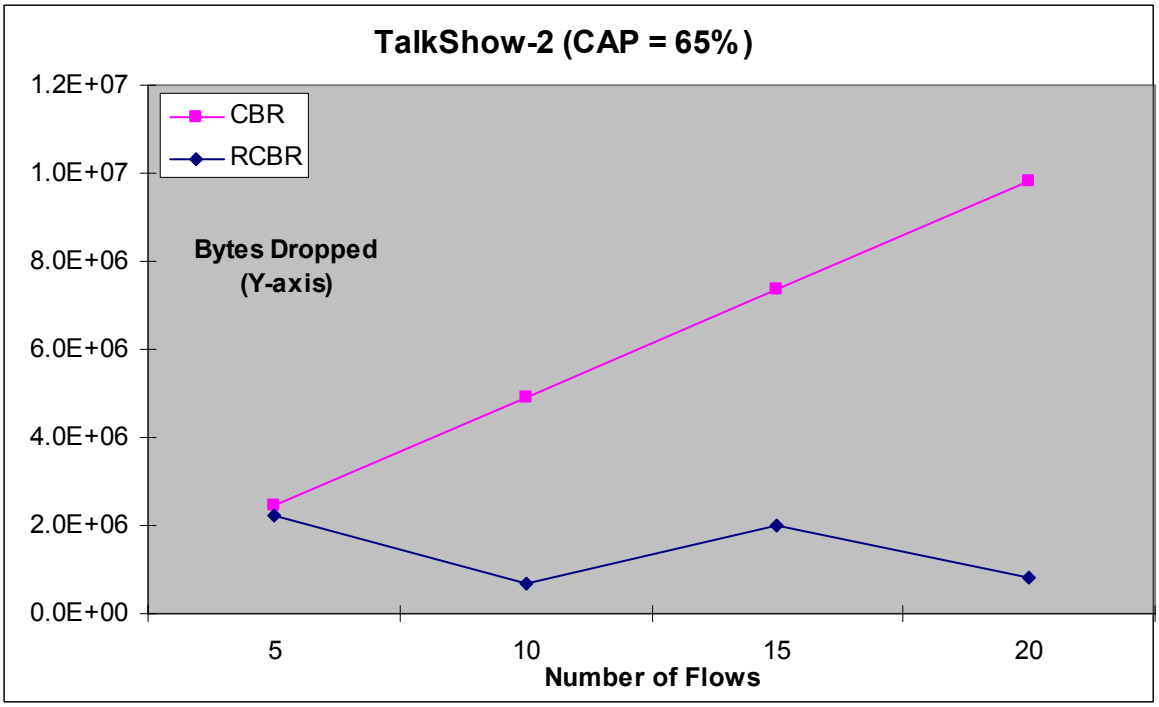


Figure 5.6: Comparison of CBR and RCBR mechanisms for 'TalkShow-2'

(vi) **Statistics and plot for the high bursty video 'Asterix'**

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	462650	55934875	16994684	10.26%	3.118
10	925300	111869750	1440401	10.26%	0.132
15	1387950	167804625	303597	10.26%	0.0186
20	1850600	223739500	79292	10.26%	0.0037

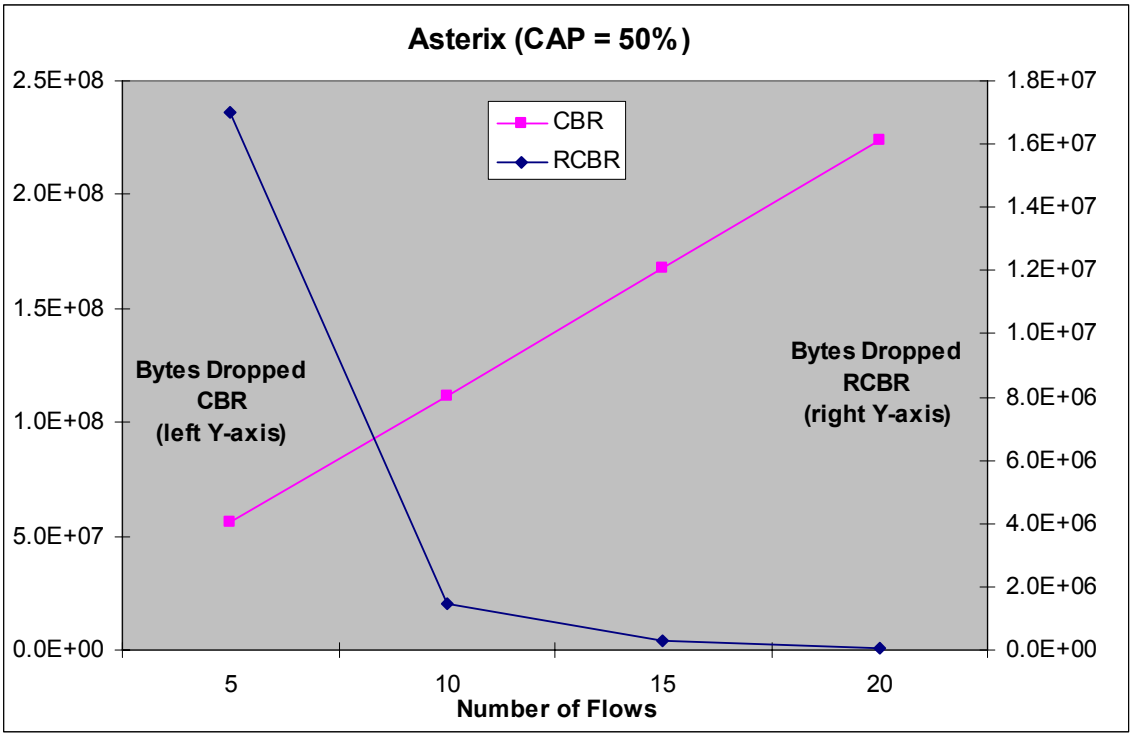


Figure 5.7: Comparison of CBR and RCBR mechanisms for 'Asterix'

Note the presence of two Y-axis scales, left for CBR and right for RCBR

(vii) **Statistics and plot for the high bursty video 'Race'**

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
5	578335	86423560	4866758	11.52%	0.6489
10	1156670	172847120	28608897	11.52%	1.9072
15	1735005	259270680	4613117	11.52%	0.2050
20	2313340	345694240	2881826	11.52%	0.0961

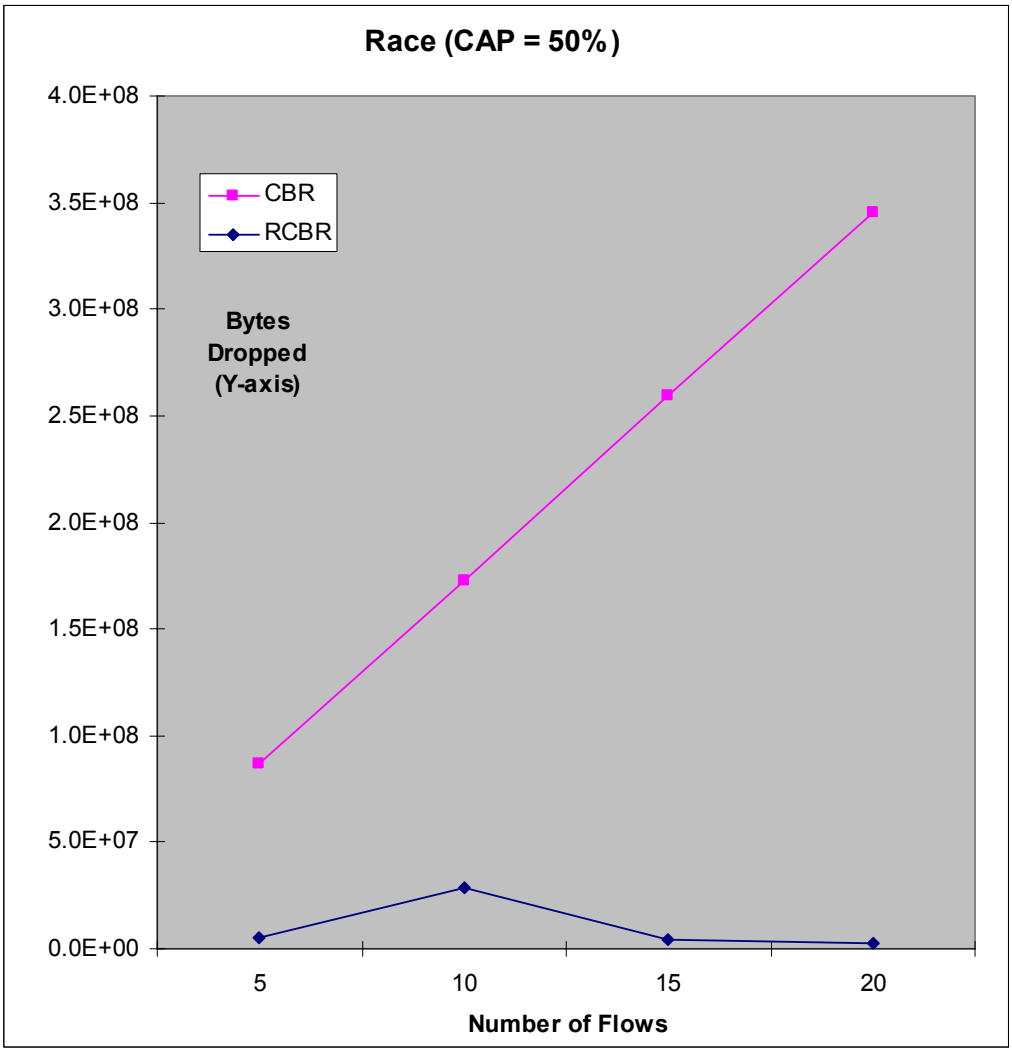


Figure 5. 8 Comparison of CBR and RCBR mechanisms for 'Race'

(viii) **Statistics and plot for a combination of different videos**

In this case a combination of videos were run together. Each iteration was done with the flow list containing a multiple of each video. The videos were 'Starwars', 'Soccer', 'Terminator' (50%), 'TalkShow-1', 'TalkShow-2', 'Asterix' and 'Race'.

# of flows	BW-Reserved	Packet Loss (CBR)	Packet Loss (RCBR)	CBR_LOSS	RCBR_LOSS
	Bytes/sec	Bytes	Bytes	%	%
7	4080993	53473501	2664137	7.90%	0.401
14	8161986	106947002	1158017	7.90%	0.087
21	12242979	160420503	483951	7.90%	0.0243
28	16323972	213894004	389923	7.90%	0.0148

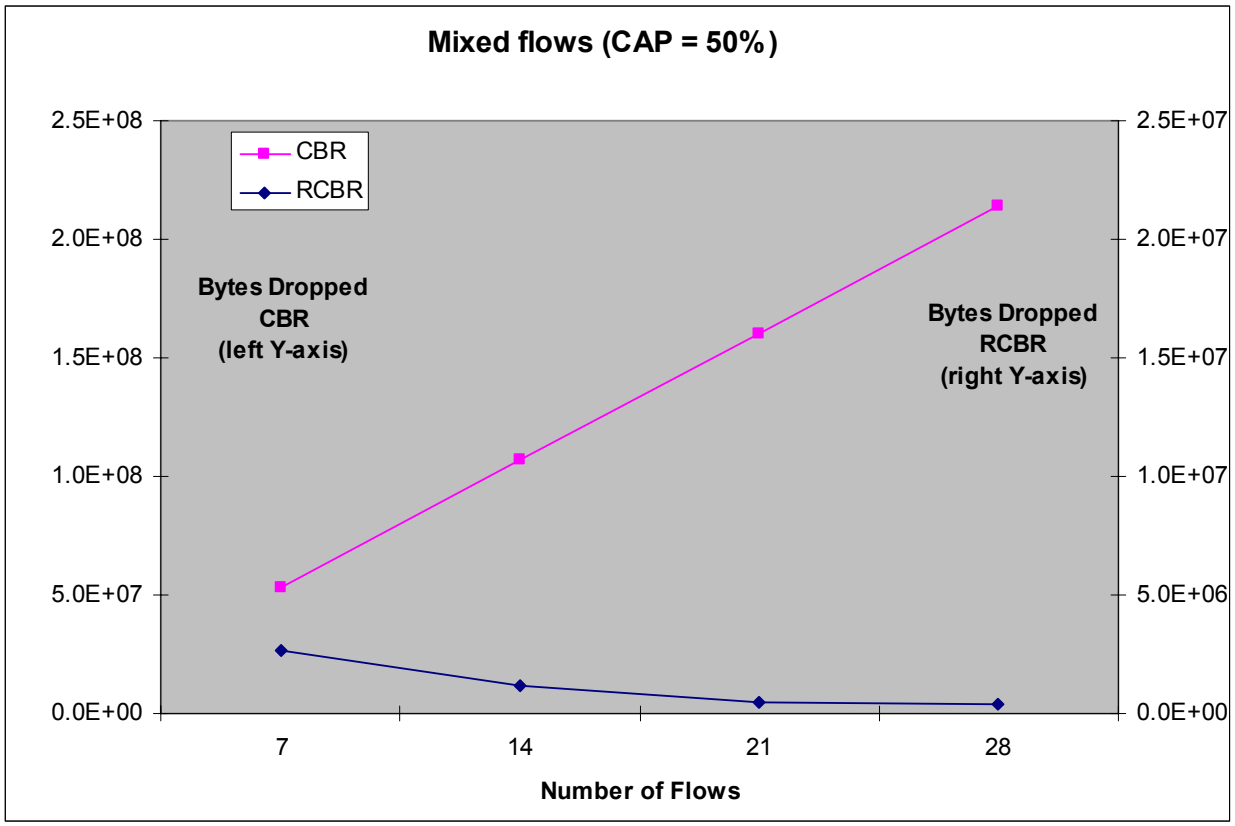


Figure 5.9: Comparison of CBR and RCBR mechanisms for 'mixed flow types'
Note the presence of two Y-axis scales, left for CBR and right for RCBR

5.4 Analysis of RCBR vs. CBR performance plots

Figures 5.1 – 5.9 depict the performance comparison of video streams working with CBR and RCBR mechanisms. The seven selected bursty flows were chosen as to show a reasonable packet drop rate (about 10% of file size) while working with a cap of 50%. Less bursty videos such as 'Mr.Bean' and 'Lambs' were omitted from our results as they showed very little packet loss for CBR even at a cap of 50% and the RCBR packet loss was too low to depict contention amongst other flows. It is worth mentioning that we assume that the flows have been admitted to the network before measuring their packet losses. Bandwidth allocation is done on a per window basis depending on the state of resources at the beginning of a RCBR window. In actuality admission control for flows is done taking into consideration the bit-rate variations over the entire playback duration. The RCBR mechanism works on the statistical multiplexing principle where the excess bandwidth released by flows during low screen activity portions is made available for the other flows. For most of the videos RCBR showed considerable reduction in lost packets.

General trends observed are:

1. Increase in the number of flows improves RCBR performance (i.e. decreases packet loss). Main reason is the statistical multiplexing gain obtained from multiple flows. Increasing the number of flows also increases the total available bandwidth which equals (**N*C* BW_{CBR}**)

   → Note that the plots show cumulative packet losses for all the flows added together. The per-flow packet loss percentages shown in the table show a stronger declining trend with increasing number of flows.

2. In certain cases RCBR shows comparable packet loss than CBR. Such phenomenon is seen with lower number of parallel flows (5 flows in our case). An intuitive reason is low statistical multiplexing gain with less number of flows. Such behavior may also be due to repeated alignment of highly bursty RCBR

windows together at the window boundaries when flows renegotiate QoS levels. Since the SBM serves one request at a time, the flows that request bandwidth at an earlier instant in the RCBR window boundary get to be allocated entire 100% of their demand while later admitted flows have to suffer lost packets if ABW mechanism is unable to adapt to the available bandwidth. Another reason may be the periodicity of bursty windows causing synchronization with other flows, which repeatedly causes the bursty window alignment phenomenon. However, in a continuous VoD server, flows admitted in the system later than other will not suffer because earlier flows would expire and later ones would move up the queue to be ahead of the sequence.

3. Mixed group of flows with different amounts of burstiness show a good level of statistical multiplexing gains and RCBR packet losses are low and similar to a flow with a medium level of burstiness.

4. Even though we change the classification of flows using their burstiness instead of their genres, it is hard to follow a strict trend in their behavior towards RCBR. A lot depends on the timing of new flows joining the existing ones on the network so that the bursty windows of some flows can benefit while being statistically multiplexed with the less bursty windows of other flows. This can be categorized as a random process but generally speaking the performance of RCBR mechanism supercedes that of the CBR mechanism.

5.5 Usage of Adaptive Buffer Window Mechanism

This section shows some statistics related to how frequently the Adaptive Window Mechanism as discussed in section 4.3.4 is used during the playback duration of the movies ‘Race’ and ‘Asterix’. The statistics collected are for the cases when the smoothing window size changed from its default value (which is 150 frames in our case) to a higher value in order to bring down the effective sending rate for that particular RCBR window. The plots are shown in figure 5.10(a) – (b) below.

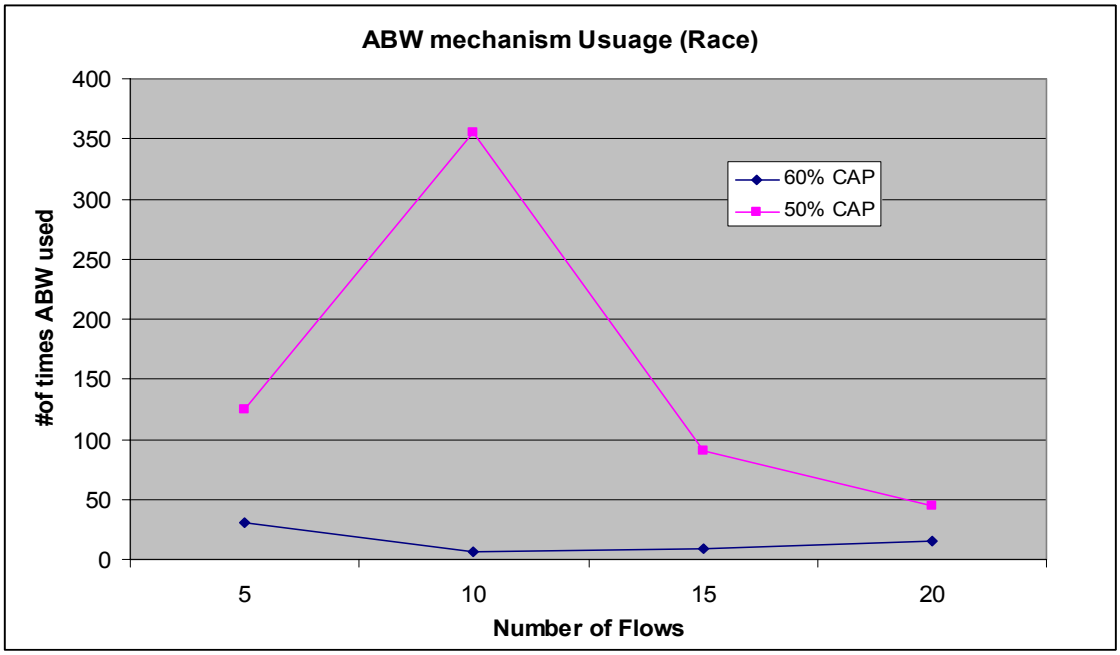


Figure 5.10(a): ABW mechanism usage statistics

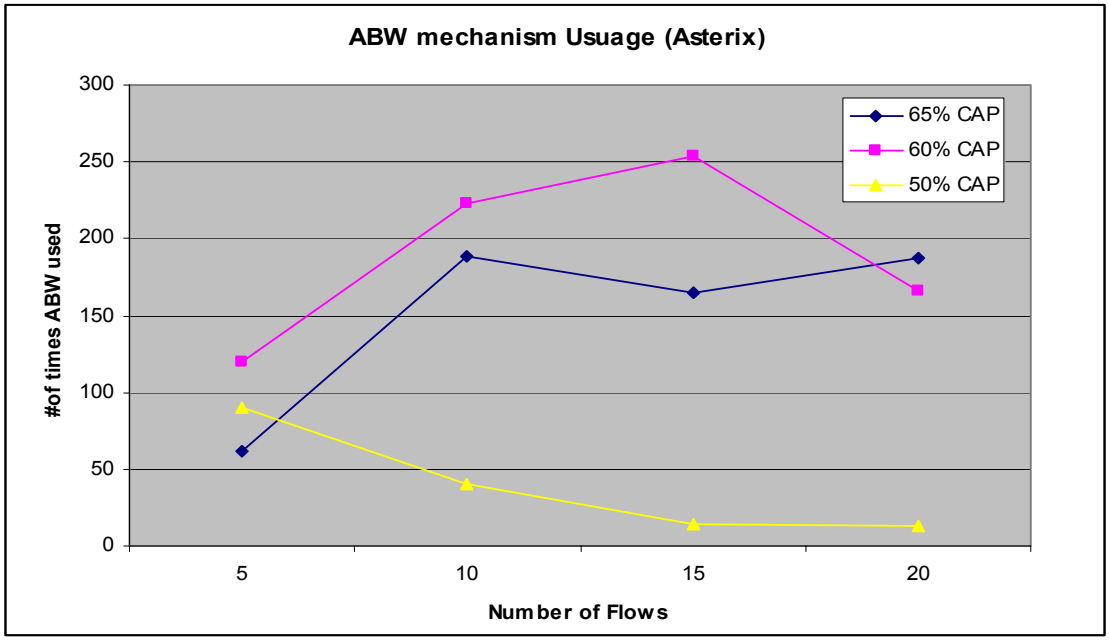


Figure 5.10(b): ABW mechanism usage statistics

5.5.1 Analysis of ABW Usage Plots

As can be seen from the plots in figure 5.10(a)-(b), the frequency of Adaptive Window Mechanism usage does not follow any fixed pattern. ABW system is best used when available bandwidth is less than but close to the requested bandwidth of a flow. This would enable a flow to use the provided flexibility in its buffer size to dynamically adapt itself to use the available QoS levels. On the other hand the ABW mechanism is not used when bandwidth contention is so high that flows are totally denied their requested QoS guarantees even while operating at the maximum buffer size; neither is it used when superfluous bandwidth is available for flows so they have no motivation to choose a larger buffer size at the cost of burdening the packet shaper and packet scheduler modules of the VoD server. As an example we have considered the usage statistics of ABW mechanism for sending 5, 10, 15 and 20 parallel flows of the videos 'Race' and 'Asterix' and considered cap sizes of 50%, 60% and 65% to show the variation amongst them

A higher usage of the ABW mechanism indicates a lower difference between the available and requested bandwidths. In the very bursty movie 'Race', the peak usage occurs at 50% cap of the CBR data rate. This shows that choosing a 50% cap would ensure a better quality-efficiency product [section 2.5.1] for the network. The trend for different number of flows can be seen in the plots. For another bursty movie 'Asterix,' we observe that setting a 60% or 65% reservation cap would be better than choosing a 50% cap because at a 50% cap the difference between available and requested bandwidths is too low to be used by the ABW mechanism. In general since we observe that performance of RCBR is lower for lesser number of simultaneous flows, ABW mechanism is also used less often. Here this is again attributed to the reduced gap between available and requested bandwidths. However the general trend of ABW mechanism being used less frequently for high number of flows (say 20 flows) can be caused by the large gap between available and requested bandwidths. The reason here is the high statistical multiplexing gain obtained by sending a high number of simultaneous flows.

5.6 Discussion

In this chapter we considered seven flows with medium to high level of burstiness and made them run with parallel copies of each other separated in time by multiples of our chosen RCBR renegotiation interval (30 seconds) so as to align their window boundaries together. We noticed that, in general, the performance of RCBR mechanism is much superior to that for CBR based systems.

RCBR relies on statistical multiplexing gain, which reaches a maximum when window boundaries of the flows are aligned together. In general performance of RCBR improves with an increase in the number of flows. Burstiness over short time scales may lead to more packet loss for RCBR and burstiness over long scales is handled well by it. Seldom RCBR may lead to a comparable (but not quite better) performance to CBR especially in the case with less number of flows. This can be attributed to bursty time windows of multiple flows aligning together, and, flows earlier in the SBM queue getting allocated 100% their requested bandwidth without the need of increasing their buffer size. This situation has a low probability of occurring but may lead to some flows at the bottom of the queue being deprived of their share of bandwidth and thus increasing the average packet loss statistic. This is not a long-term effect, as these flows do not always remain at the bottom of the queue. They move up the stack when (1) earlier flows expire and new flows are added at the bottom of the stack, (2) when flows earlier in the stack decrease their bandwidth requirements, the bottom level flows can use the released bandwidth and for subsequent windows get an advantage of already been assigned a higher bandwidth level.

The Adaptive Buffer Window (ABW) mechanism was used in all the RCBR implementations with the smoothing window file. This mechanism is best and most frequently used when the difference between required and available bandwidths is less. Its usage statistic can be useful for determining a good quality-efficiency product for the network implementing QoS so as the get the optimal performance without over-provisioning of any resources.

CHAPTER 6 – Summary and Future work

This research aims at exploring an affordable and easy to implement technology for streaming video servers. Video-on-Demand is a fast developing service and holds tremendous consumer potential. Since the last mile bandwidth has now evolved from slow connections to broadband Internet, resource intensive services are gaining much popularity and providing the motivation for further research into this field. Traditional VoD setups used analog coaxial cable networks to send digital video to their subscribers. These were rarely shared and did not carry other crucial traffic such as Internet data and voice. Quality of Service mechanisms were developed to solve bandwidth contention issues by providing stringent guarantees to paid subscribers. Conventional implementations of QoS date back a few years suffer limitations imposed by vendors and security hazards. Compatibility issues prohibit wide scale deployment of resource reservation protocols such as RSVP over public networks. However QoS mechanisms can easily be used within private campuses such as university campuses, industry and neighborhood networks to support streaming video on demand services for educational, training and entertainment purposes.

6.1 Summary of Analytical Results

Our initial motivation to proceed with this research was the conventional implementation of the Resource Reservation Protocol (RSVP) to provide Quality of Service. It popularly exists as a per-flow mechanism to establish a constant reservation state along the entire route between the sender and the receiver of a data stream. The constant bit rate reserved provided a convenient way of admission control at the cost of over-provisioning network resources. Renegotiated Constant Bit Rate (RCBR) service proposed dividing the playback duration of a video into small sized windows each with its own bandwidth level. Through our analysis as discussed in chapter 3, we compared the performances of RCBR and the conventional CBR mechanisms. Our theoretical results showed that RCBR is a very befitting mechanism for sending multimedia traffic over a network. RCBR showed high amounts of bandwidth savings for less bursty videos where the ratio

of peak – average data rate was very high. The performance of RCBR increases as it scales to multiple flows. The extra bandwidth released by flows become available for the rest others. RCBR mechanism has the compatibility to work on top of other QoS mechanisms such as those specified under IETF's Differentiated Services. RCBR can also work on top of buffering mechanisms if they lead to varying data rates in their playback duration. Although we presented the results for all practical values of RCBR window sizes, we recommend using RCBR with a time window equal in size to a multiple of the QoS refresh period, which typically is 30 seconds.

6.2 Summary of test-bed implementation

Our motivation of implementing a test bed was to study the performance of RCBR on a real world network unlike the simulations and analysis done so far by other researchers. We implemented the test bed with RSVP as our choice of a QoS mechanism. Our server and client applications were implemented as multi-threaded on individual but separate hosts and were connected through a router by means of a 100Mbps fast Ethernet connection. We concentrated on the networking performance of such a system and not on the user interface so there was no MPEG decoding and display algorithms implemented.

We used the concept of application level buffering where we introduced the concept of dividing each RCBR window into multiple smoothing windows so that data sent in each smoothing window would be sent at its mean rate and using mean sized packets. The advantage achieved by this mechanism was not to over burden the VoD server's packet shaper and packet scheduling modules and prevent them from causing scalability issues and performance bottlenecks. We also proposed the Adaptive Buffer Window (ABW) mechanism that made use of the Resource Tracking Agent (RTA) along with the Smoothing Window File to provide flexibility to each flow to adapt (lower) its bandwidth requirements incase the available bandwidth is insufficient. This would be done at the cost of using the packet shaper resources of the VoD server.

6.3 Summary of Experimental Results

With the experiments we aimed to compare the performance of different flows implementing CBR and RCBR in terms of data loss during the entire duration of file transfer. We put limiting constraints on the bandwidth allocated for CBR flows to cause packet drops when the data stream bursts exceed the limiting value. The same constraint was also imposed on identical combination of flows but using the RCBR implementation. We observed that the packet losses for flows following the RCBR system were much lower than those following the CBR mechanism. RCBR. We considered flows with varying level of burstiness and observed that the general performance of RCBR improves with an increase in the number of flows. This may be attributed to the statistical multiplexing gains achieved by the flows.

Seldom RCBR may lead to a comparable (instead of better) performance to CBR. This happens mostly with less number of parallel flows. Besides the low statistical multiplexing gain, another reason can be the alignment of bursty RCBR windows of all flows which may lead to flows lower in the SBM (FIFO) request queue being deprived an equal share of available bandwidth. In this case they need to use the Smoothing Window Mechanism to see if they can change their sending buffer to fit into the available bandwidth. These flows don't always remain lower in the SBM queue because (1) earlier flows expire periodically pushing them up the stack and (2) earlier flows decrease their bandwidth request thus releasing resources available for use by all other flows.

The Adaptive Buffer Window (ABW) mechanism was used in all the RCBR implementations with the Smoothing Window File (SWF). This mechanism is most frequently used when the difference between required and available bandwidths is less than a lightly loaded network. Its usage statistic can be useful for determining a good quality-efficiency product for the network implementing QoS so as the get the optimal performance without any over-provisioning of resources.

6.4 Suggestions for Future Work

In this research work we have produced empirical results by implementing a true RCBR model on a test bed network. Our analytical results indicated the potential of saving bandwidth and increasing the network utilization using RCBR. We did not work with the same performance metrics for theoretical and experimental analysis because examining the flows' behavior under a stick bandwidth constraint was our aim. Some more analysis and research needs to be done before this technology can be widely implemented in commercial Video-on-Demand servers. Here are some suggestions for future work:

1. Testing of an RCBR implementation over a wide area network across public routers carrying general background traffic. This would need a proper understanding of Subnet Bandwidth Management functionality over multiple subnets.
2. Use of RSVP's public distributions over UNIX based systems such as KOM-RSVP engine
3. Connecting empirical results to simulation results using OPNET, QualNet or NS2 to study scalability, compatibility and failure issues
4. Implementing MPEG decoders and players on client side software to study the visual quality effect while comparing CBR and RCBR mechanisms
5. Implementing an entity for generating MPEG trace files and Smoothing Window Files using a video file instead of generating them while encoding
6. Implementing a network entity that acts as a database of trace files and smoothing files to serve VoD servers with updated statistics, versions and newer videos
7. Implementing the RCBR system for other file encoding algorithms such as MPEG-4, MPEG-7, ASF®, Real Media® and others.
8. Research on a suitable generic protocol to implement the initial client server handshake for VoD, network communication with the RTA and communication with the trace-file and smoothing window file server (if they exist over the network)

6.5 Thesis Summary and Final Thoughts

Our motivation for carrying out this research was to test the RCBR mechanism for Video-on-Demand systems. We showed it to be superior to the currently implemented CBR mechanism Since QoS mechanism are not supported by default over the public Internet, we focused more on the benefits of campus wide networks for the VoD technology. This service has great potential since the future networks with IPv6 and unrestricted bandwidth would encourage IP based Televisions and entertainment devices. More research would be required to implement RCBR for commercial systems such as neighborhood cable TV or university campus networks for educational use. This research provides a stepping stone closer to the same.

References

[IEEE8021P] "Information technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Common specifications - Part 3: Media Access Control (MAC) Bridges: Revision (Incorporating IEEE P802.1p: Traffic Class Expediting and Dynamic Multicast Filtering)," ISO/IEC Final CD 15802-3 IEEE P802.1D/D15, November 24, 1997.

[CLP00] Carter S.W., Long D., Paris J., "An Efficient Implementation of Interactive Video-on-Demand," *MASCOTS*, 2000, San Francisco, California, pages 172-179.

[CK96] Chang K., Kung H. T., "Efficient Time-Domain Bandwidth Allocation for Video-on-Demand Systems," *Fifth ICCCN*, 1996, Washington D.C., pages 68-77.

[CH93] Cohen D.M., Heyman D.P., "A Simulation Study of Video Teleconferencing Traffic in ATM Networks," *IEEE INFOCOM*, 1993, San Francisco, California, pages 894–901.

[DLG91] D.L. Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of ACM*, vol. 34, no. 4, 1991, pages 46–58.

[DS97] J. K. Dey, S. Sen, J. Kurose, D. Towsley, and J. Salehi., "Playback Restart in Interactive Streaming Video Applications," *International Conference on Multimedia Computing and Systems*, 1997, Ottawa, Canada, pages 458-465.

[FJS97] Feng W., Jahanian F., Sechrest S., "An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video," *Multimedia Systems*, vol. 5, no. 5, 1997, pages 297–309.

[MPGURL] MPEG Video Group Homepage, http://wwwam.hhi.de/mpeg-video/#MPEG_General (URL last visited November 2003)

[KLG94] Gong K. L., "Berkeley MPEG-1 Video Encoder, User's Guide," University of California, Berkeley, Computer Science Division-EECS, 1994.

[KT99] Kantarci A. and Tunali T., "Transmission of Multimedia Data Over IP," *Fourth Symposium on Computer Networks*, 1999, Istanbul, Turkey, pages 68-76.

[KT97] Krunz M., Tripathi S., "Exploiting the Temporal Structure of MPEG Video for the Reduction of Bandwidth Requirements," *IEEE INFOCOM*, 1997, Kobe, Japan, pages 67-74.

[KK03] Kuo G.S., Ko P.C., "Dynamic RSVP Protocol, " *IEEE Communications Magazine*, vol. 41, no. 5, 2003, pages 130-135.

[LCY94] Lam S, Chow S, Yau D., "An algorithm for Lossless Smoothing of MPEG Video," *ACM SIGCOMM* , 1994, London, U.K., pages 281-293.

[MSD97] M. Grossglauser, S. Keshav, and D.N.C. Tse, "RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic," *IEEE ACM Transactions on Networking*, vol. 5, no. 6, 1997, pages 741-755.

[MWG93] M.W. Garrett, "Contributions Toward Real-Time Services on Packet Switched Networks," Ph.D. dissertation, Columbia University, 1993.

[MM01] Menth M. and Martin, R.: "Performance Evaluation of the Extensions for Control Message Retransmissions in RSVP," *7th International Workshop on Protocols For High Speed Networks*, 2002, Berlin, Germany, pages 35-49.

[ROS97] Rose O., "Traffic Modeling of Variable Bit Rate MPEG Video and its Impacts on ATM Networks," Ph.D. dissertation, Wurzburg, 1997.

[PE92] Pancha P, El Zarki M., "Prioritized Transmission of Variable Bit Rate MPEG Video," *IEEE GLOBECOM*, 1992, Orlando, Florida, pages 1135–1139.

[RRB93] Reininger D, Raychaudhuri D, Melamed B, Sengupta B, Hill J., "Statistical Multiplexing of VBR MPEG Compressed Video on ATM Networks," *IEEE INFOCOM*, 1993, San Francisco, California, pages 919–926.

[SZ96] James D. Salehi , Shi-Li Zhang , Jim Kurose , Don Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing," *IEEE ACM Transactions on Networking*, vol. 6, num. 4, 1998, pages 397-410.

[SJ93] Stone, D.L., and K. Jeffay. "Queue Monitoring: A Delay Jitter Management Policy," *4th International Workshop On Network and Operating System Support for Digital Audio and Video*, 1994, Lancaster, U.K., pages 151-162.

[BER] Bernet Y. (2000, Nov 14). *Networking Quality of Service and the Windows Operating System*. (1st edition), Sams publishing.

[ZFV92] H. Zhang, and D. Ferrari, D. Verma, "Delay Jitter Control for Real Time Communication in a Packet Switching Network," *IEEE Tricomm*, 1991, Chapel Hill, North Carolina, pages 35-43.

Glossary of Acronyms

ABW : Adaptive Buffer Window

ACS :Admission Control Service

API :Application Programming Interface

ASF :Advanced Streaming Format

ATM :Asynchronous Transfer Mode

BW :Bandwidth

CBR :Constant Bit Rate

DCT :Discreet Cosine Transform

DiffServ: Differentiated Services

DNS :Domain Name System

DSBM :Designated Subnet Bandwidth Manager

DSCP :DiffServ Code Point

FIFO :First In First Out

GOP :Group of Pictures

GQoS :Generic Quality of Service

GUI :Graphical User Interface

HTTP :Hyper Text Transfer Protocol

IETF :Internet Engineering Task Force

IntServ:Integrated Services

IP :Internet Protocol

JPEG :Joint Pictographic Expert Group

LAN :Local Area Network

MAC :Medium Access Control

MB :Mega Byte

MPEG :Moving Pictographic Expert Group

MPLS :Multi Protocol Label Switching

NHOP :Next Hop

NS2 :Network Simulator –2 (Simulator)

OPNET: Optimum Network (Simulator)

PATH : RSVP path message

PCBR :Piecewise Constant Bit Rate

PHB :Per Hop Behavior

PHOP :Previous Hop

QoS :Quality of Service

QOS-SP : Quality of Service - Service Provider (Microsoft Specific)

RCBR :Renegotiated Constant Bit Rate

RED :Reservation Establishment Delay

REFRESH: RSVP Refresh messages (PATH + RESV message exchange)

RESV :RSVP reserve message

RFC :IETF's Request for Comment (document)

RSVP :Resource Reservation Protocol

RTA :Resource Tracking Agent

RTCP :Real Time Control Protocol

RTD :Reservation Teardown Delay

RTP :Real Time Protocol

SBM :Subnet Bandwidth Manager

SW :Smoothing Windows

SWF :Smoothing Window Format

TCMON :Traffic Control Monitor (tool)

TCP :Transmission Control Protocol

UDP :User Datagram Protocol

VBR :Variable Bit Rate

Vitae

Mukul Kishore was born on September 2, 1978 in the city of New Delhi, India. He earned his Bachelors Degree in Electrical Engineering from Delhi College of Engineering affiliated with the University of Delhi in 2001. He then joined the Bradley Department of Electrical and Computer Engineering Fall 2001 for the Masters of Science in Electrical Engineering. He spent his entire graduate school term at the Alexandria Research Institute based in Old Town Alexandria, where he worked as a Graduate Teaching Assistant for his department.

Mukul has been a student member of IEEE since 1997 and has participated in activities of IEEE's student chapters at Delhi College of Engineering and Virginia Tech. He has also held student memberships of Society of Automotive Engineers and Society of Computer Engineers.