INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

VIDEO TRANSMISSION OVER CONSTRAINED BANDWIDTH CHANNELS

Girish Mahanta

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

October 1997

Abstract

Video Transmission over Constrained Bandwidth Channels

Girish Mahanta

Multimedia applications are often considered adaptive in the sense that their behaviour can be *adapted* to the available resources. One way to achieve this is through *selective dropping* of media frames. Different media types have different tolerance to loss, for example, humans are more sensitive to loss of continuity in audio as compared to video. It is, therefore, attractive to develop best effort services targeted for a particular media type (e.g., video). Such a service should provide *graceful performance degradation* as resources become scarce by exploiting the particular characteristics of the media type.

In this thesis, we consider providing a best effort service targeted for video transmission, by multiplexing multiple video streams over a reserved bandwidth network channel. The emerging services architecture in ATM networks as well as the Internet provide the capability to reserve resources such as bandwidth, and to use them to provide higher level services. We use simulation to consider the effect of different scheduling and buffer management policies on the *performance* when the bandwidth is scarce and losses are inevitable. By using effective frame rate as a performance measure, we show that FIFO scheduling is unable to ensure fairness in terms of equitable performance degradation of the multiplexed streams. Our simulations reveal that explicit bandwidth allocation is required, and that this bandwidth allocation must be (1) targeted to ensure that the performance of each of the multiplexed streams degrades equitably, and (2) must be dynamically changed to take advantage of the *time varying characteristics* of video data.

Acknowledgments

I am indebted to my supervisor Dr.Manas Saksena, for his guidance and constant support all through the course of this work. His patience and understanding were helpful in the completion of this thesis. I acknowledge with thanks the financial support I received through Dr.Saksena.

I extend my thanks to Dr.J.W.Atwood, Dr.F.Khendek for being on the examining committee and Dr.L.Narayanan for chairing the defence.

I would like to express my gratitude for my parents and sisters, for their love and affection.

I wish to thank the fine group of systems analysts for their timely solutions during the cource of my Master's programme.

Thanks to Mr.Shawn D., Ms.Arbinder K. Pal and Ms.Jaya N. for their help during proof reading of the thesis.

I am grateful to Mr.Krishnamurthy W., Mrs.Savitha G.C. and Mr.Prakash S. for continually urging me to complete the thesis. I thank Mr.Narayan S., Mrs.Savita D.C., Ms.Sunita H.K., Mr.Veeresh K.M. and Mr.Surya Prakash N. for being in touch all the time and never missing me out on anything. Also, my thanks to Mr.Mahesh B., Mr.Venkateshwara Rao A., Mr.Vijay K. and Mr. Seetharaman S. for making my stay in Montreal, a pleasant one.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent technological advancements in broadband communications, computers, and media peripherals have led to the emergence of a new paradigm of computing in which continuous media (e.g., video and audio) is treated in much the same way as discrete media (data). This paradigm shift is reflected in distributed multimedia applications such as virtual reality, media-on-demand, video conferencing, tele-marketing, tele-medicine, etc., that seek to revolutionize the computing world. It is commonly accepted that these applications will use a single integrated network to carry a wide variety of data (video, audio, images, text, etc.) with processing support at end systems (i.e., host computers) connected to a variety of multimedia peripherals (speakers, microphones, television monitors, camera, etc.).

In a typical distributed multimedia application, data is generated at a source computer, transported over the communications network, and finally processed and presented at the destination computer. The successful use of multimedia applications requires that multimedia data is presented with high *perceptual quality* to the user. The perceptual quality of a multimedia presentation depends on maintaining the temporal relationships between media items. Such relationships include, for instance, displaying video frames one every 30 ms or so, maintaining "lip-synch" between audio and video, etc. Furthermore, live multimedia applications also require that the end-to-end latency in presenting the data is small (typically less than 200 ms). The timeliness requirements are soft, in the sense that small deviations do not affect perceptual quality. On the other hand, consistent large deviations from the desired temporal behaviour is unacceptable and renders the presentation useless.

Most multimedia applications, unlike traditional data, can tolerate some data loss

without resulting in the transmitted data being completely futile. On the contrary, traditional data require an error-free transmission. In addition, multimedia applications can exhibit graceful degradation in performance under controlled losses by producing reasonable quality outputs at the receiver end. Also, multimedia applications are adaptable to such an extent that they can experience some variation in service without affecting the quality. Such a behaviour can be seen in the case of a movie where the video segment can lag the audio segment up to 40 ms without being noticed by the user.

A major challenge in the development of next generation computer networks and systems is the incorporation of resource management policies that cater to the specific needs of distributed multimedia applications. Clearly, this requires a network to offer services beyond the "best-effort" services offered in today's Internet. Recent developments in computer networking research suggest a convergence towards a two-pronged approach to this problem.

- At the lower layer, the computer network provides generic services using which an application can reserve resources, and expect some quality of service guarantees provided it conforms to a traffic contract. For example, an application may request a Constant Bit Rate (CBR) service that guarantees a bandwidth B to it. The network on its part commits that all packets will be sent unless the bandwidth usage at any time instant is greater than to the negotiated bandwidth B.

- Using such generic services provided by the computer network, the higher level services tailored to a particular media type such as video or a particular application such as video presentation, determines how to use the resources available to provide the quality of service desired by the application.

1.1 Video Transmission

In this thesis, we address video transmission as part of higher level services that can be used in many applications. Video data in the uncompressed form requires high bandwidth for transmission and thus calls for compression. The different amounts of compression achieved produces frames of uneven sizes and thus results in time varying data rates. Furthermore, video can suffer from small amounts of loss without

substantially degrading the quality. In case of a major short-fall in a resource such as bandwidth, the video can exhibit graceful degradation by employing frame level drops. The time varying nature of video data prompts an idea wherein a bandwidth reservation can be shared for transmission of multiple video streams so that more effective use of link capacity is possible. We thus choose to study the transmission of multiple video streams over a reserved bandwidth.

Networks providing resource reservation facilities consist of objects such as sources, sinks and routers, which employ buffers to manage dynamic data rates of media such as video. Due to time varying data rate, the required bandwidth typically differs from that of reserved bandwidth by a small quantity. If such a variation results in higher demand for bandwidth, then it is offset by storing the excess data in buffers and thus bridging the gap between asking and reserved bandwidths. Data in the buffers is governed by a scheduling discipline, which decides on what data to send and when to send it. The number of buffers employed is determined such that the delay due to queueing is within real-time requirements. Hence, the extent to which buffers can be employed is limited by the permissible delay.

There has been considerable research towards exploring the transmission of multiple video streams to determine the bandwidth and buffer requirements for zero loss. The bandwidth in this case is the average data rates of all streams being transmitted. On the other hand, we find relatively less work in case of inadequate bandwidth. The problems will be acute, in case a vital resource such as bandwidth is short of required quantum. Existing networks can result in insufficient bandwidth for video applications, which are on the higher end of resource requirements. As a consequence of insufficient bandwidth, data loss is certain due to over-flow of permissible buffer space. Such drops are regulated by buffer management policies which decide when the data are to be dropped. Data drops so inflicted lead to random information loss and thereby adversely affect quality of video. A possible solution for this problem is to eliminate random losses. This has resulted in proposing frame level drops. If a particular video frame consisting of many packets is found to incur random losses, then instead of sending an incomplete frame, it is better to drop the entire frame. Drops are forced this way to prevent bandwidth from being consumed by incomplete video frames. Such drops will help increase the number of intact frames sent resulting in improved visual quality. Thus, we have selected constrained bandwidth channels for our study.

We attempt to control the loss of video data in order to ensure graceful degradation in the performance under reduced bandwidth conditions. The losses so experienced cannot be expressed in terms of percentage of packet losses as can be done with traditional data, since video data is subjective in nature. Thus, we need measures capable of capturing the visual quality for a single stream. In order to accomplish this, we employ a measure called *Effective Frame Rate.* This metric can be used to loosely quantify the loss in quality due to frame drops against the original stream. We opt for frame level drops to limit the scope of the thesis, though slice or macroblock level drops would have resulted in better performance. Furthermore, we focus our solutions to obtain a low variation of Effective Frame Rate so that the fluctuations in quality are not too high.

As far as the performance of multiple video streams is concerned, we want to ensure fairness among sequences. This is achieved by maximizing the minimum quality among all streams leading to best possible quality video at the sink.

1.2 Related Work

In this section, we briefly describe the related work connected with video transmission under insufficient bandwidth. Typically from such a transmission we expect the best possible quality video at the receiver end. The video data somehow has to be controlled to meet the desired quality. Research in this direction shows two main categories. The first category concerns dropping video data. For this purpose video data components which are readily available (which do not need processing) are employed. Examples of such components include slices and frames. The second category focuses on minimizing the bandwidth requirement of video streams by manipulating one of the video generation parameters at the encoder. One such example is the quantizer value which can be used to effectively bring down the data rate. We review some projects covering both these approaches. In addition, we also note the application of video gateways to exclusively manage video streams in the current network environments.

1.2.1 Optimal Video Transmission

The work by [KMS95] is driven by the need to address video transmission of a single stream over insufficient reserved bandwidth link. The authors address optimal transmission of prioritised data that is a characteristic of MPEG video sequences and thus is of interest to us. Their goal is to maximize the visual quality of video at the sink. The model proposed by them uses the frame size as weight. Their algorithms appear in two versions: the first version assumes that the exact channel bandwidth is known, while the second version assumes that the bandwidth is uniformly distributed in an interval of known endpoints. The premise of their algorithms is to minimize the expected maximum gap due to unplayable or dropped frames. The algorithms given by them are less suitable for real-time applications like video conferencing, as the frames need to be reassembled for playing in addition to the regular decoding delay.

1.2.2 Filters for Rate Control

Another related work at Lancaster [YMGH96] deals with transmission of a single video stream under limited link capacity. It aims at implementing a dynamic quality of service control led architecture capable of optimizing a variety of requirements evident amongst the large group of simultaneous users. This work takes into account the heterogeneity in terms of hardware, software and network protocols among others. The authors propose filters, which employ certain heuristics to reduce bandwidth requirements with controlled loss of quality. The filters-governed data rate control employed in this work is of relevance to our work. The following are the six types of filters developed by them:

- **Codec filter.** This is used to compress or decompress a bitstream, usually by transcoding between two compression standards. Implementation of this filter may or may not need full decompression and recompression depending on the standard used.

- **Frame-Dropping Filter.** This filter drops frames thereby considerably reducing the data rate. In case of MPEG video, this filter has the knowledge of the type of frame and uses this knowledge while dropping according to their

importance. For instance, in an IPB video sequence of MPEG, first B, next P and finally I frames would be selected for dropping.

- **Frequency Filter.** These filters operate on the semi-uncompressed data in the frequency domain. An example of these filter's input is the DCT-coefficients. This category includes facilities to remove: (i) higher frequency DCT-coefficients, (ii) chrominance part, and (iii) all the colour information, from a stream.

- **Mixing Filter.** This filter is used to mix streams together, reflecting the multiplexer operation. The mixing can be performed at two levels, (i) at the frame level and (ii) at the slice level. Slice level mixing incurs more processing as incompatible slices need to be matched with the help of padding.

- **Re-Quantization Filter.** This filter operates on the DCT-coefficients to de-quantize them. These de-quantized coefficients are re-quantized with a higher quantization step. This process leads to substantial reduction in data requirement at the cost of quality.

- **Slicing Filter.** This filter is used to increase the number of slices in a frame. The main idea is to provide protection against data loss by not propagating the effect of some data loss beyond the particular slice. However, the addition of every slice involves an overhead of 38 to 45 bits, in terms of new slice header length.

As mentioned in the introductory section, we notice data rates being modified in varying quantities at different levels starting from frames to DCT coefficients. The previous work by [YMGH96] provided an optimal solution, while in this case we observe rate reduction to bridge heterogeneity gaps among various network components. A common aspect between this and our work is the drop entity, which is a frame.

1.2.3 Feedback Based Rate Control

In this paper, the authors [KM96] employ a feedback mechanism to monitor the data rates of multiple video streams in distributed ATM networks. In this case the network elements such as switches, routers and others inform the sources generating video data of their changing situations so that sources can adapt the data rate with the help of the encoder. The feedback is in the form of a single bit in the cell header

called *Explicit Forward Congestion Notification* (EFCN), for the source to adapt its data rate.

This paper presents two methods designed for bursty sources, which are modified for video data, and a new technique. Among the two schemes designed for bursty sources, the first by *Ramakrishnan and Jain* sets the single bit when the average queue length of the switch is greater than or equal to a particular threshold. This scheme is modified using the assumption that the congestion bit is set based on the queue occupancy value at the instant of cell transmission.

The second scheme by *Mitra and Seery*, adapts the windows or data rates by implicit feedback (derived from measured round trip delay) or explicit feedback (about queue occupancy). The switches periodically send queue occupancy information to the sources directly, with the bit set according to the instantaneous queue occupancy level. Window sizes are then set to reflect the network conditions. Kanakia and Mishra modify this method such that after setting the bit the cells are forwarded to the receiver rather than the source to suit EFCN.

The third scheme developed by the authors uses the single bit to convey more accurate information about the queue occupancy. Their premise is that the receiver is capable of capturing more information about the slowly varying quantities such as queue levels and this can be passed on to the source. One way this scheme can be implemented is by employing multiple thresholds to depict various levels of queue occupancy.

The results show that the feed-back schemes work well for all the three schemes with minor differences. But, these conclusions are accompanied by unfair services to the users. To address the fairness issue, the authors have proposed a scheme which allows the sources to adjust their bit rates so that the quality of the compressed bit-stream matches the target image quality.

This work is similar to ours from the point of view of the fairness issue and in general improving the quality of video. The main difference is that we address the fairness issue with the help of few scheduling techniques, while in this case we notice feedback being employed resulting in the transmission of additional data in the reverse path of video transmission.

1.2.4 An Application Level Video Gateway

At present multicast transmission across the Internet assumes that a fixed average bandwidth is available over the entire network. This restricts the users having higher bandwidth links from achieving a better quality. The authors [AM95] address this problem by proposing an architecture for video transmission, wherein a single transmission can be decomposed into multiple sessions with different bandwidth requirements using an application level gateway. Their video gateway manipulates the data and control information of the video sequences to establish a single logical conference from pairs of sessions. It employs transcoding and rate-control to effect bandwidth adaptation. They present an algorithm to transcode Motion-JPEG to H.261 in real-time. The Real-time Transport Protocol (RTP) is embedded into the video gateway thereby allowing it to interoperate with current video tools.

We are interested in this work as it addresses data rate adaptation through transcoding and rate control. Among these, transcoding incorporating RTP is a special feature compared to our work. On the other hand, rate control enforced over a transcoded stream with frame drops is similar to our work. A notable feature of this video gateway is the mechanism it provides to dedicatedly handle video with the help of advanced protocols like RTP and RSVP, where RSVP can be used to reserve resources along the path of transmission. The solution to our problem could be implemented on top of a video gateway such as this one.

1.3 Contributions

In this thesis, we have explored a few simple mechanisms to manage bandwidth and buffers, when multiple video streams share a bandwidth reservation. Our design space consists of: (i) whether the buffers should be shared or partitioned and (ii) whether the bandwidth should be shared and managed using a FIFO scheduler, or partitioned (perhaps dynamically) and controlled using a scheduler employing some Generalized Processor Scheduling mechanism [PG93].

Our results suggest that in order to ensure fairness among different streams, it is necessary to partition both buffers and bandwidth. A second outcome of our tests is that the specific manner in which buffers are partitioned is not so important, as long as some minimum buffering is available. Finally, our results indicate that bandwidth

partitioning should be dynamic to exploit the time varying characteristics of video sequences. Also, such a partition should be driven by "quality" requirements.

Chapter 2

Video

Video has become one of the main constituents in the emerging multimedia and telecommunication applications. This has attracted much attention leading to a spur in research directed towards integrating video into the world of computers. In this chapter, we address the various issues connected with digital video. We begin with fundamental aspects of video in the digital form. We then show the need for compression and present relevant compression techniques. Finally, we detail video standards developed for the computer domain.

2.1 Digital Video Preliminaries

Recent advances in technology are paving the way to integrate video, computer and telecommunication industries together on a single platform called "Multimedia". In order to integrate video into the multimedia platform, the video signal needs to be scalable, platform independent, robust, and error resilient. In addition, it also needs to facilitate interactivity and editing. Analog video signal unfortunately fails to meet these requirements, thus, making it unsuitable for such multimedia platforms [AB95]. Hence, switching to digital not only answers these problems, but also opens up a whole new range of sophisticated digital video processing techniques, which can improve the quality of video.

Video is a sequence of time varying images, which convey *visual* information. Conversion of video from analog to digital format starts with the process of sampling. The basic unit of sampling is called a *pixel*. The sampling process yields a set of

parameters essential to represent a digital video signal in terms of pixels per line, number of lines per frame, resolution, aspect ratio and finally frame rate. In addition to these, images in video sequences are characterized by colour spaces or chrominance and luminance. In the following subsections, we provide an overview of these aspects.

2.1.1 Resolution

Resolution specifies the matrix of pixels that constitutes the image size, also referred to as *spatial resolution* [SN95]. The process of sampling in fact establishes the resolution of an image. Sampling in the horizontal direction defines the horizontal resolution of the picture. In contrast, sampling in the vertical direction yields the vertical resolution, indicated by the number of lines. On the other hand, temporal sampling determines the frame rate of the video sequence, quoted as *frames per second* (fps). Aspect ratio is defined as the ratio of horizontal to vertical resolution of the image.

2.1.2 Colour Spaces

Video images in addition to the resolution are defined by the way colour information is stored [AB95]. A gray-scale picture has only one colour component: luminance. Since brightness is a complicated item to quantify, luminance was defined [Poy97]. For example, in an image with 8 bits/pixel scale, higher range values in this scale indicate lighter gray colour. Thus, a value 0 represents the colour black and 255 represents the colour white.

In order to capture the chrominance part of colour video images, three components are required. These components can be denoted by the most popular colour space as RGB. In this colour space, the R, G and B represent the amount of red, green and blue respectively. True-colour pictures use 8-bits for each component and thus yield 24 bits/pixel. Another known colour space is the YUV. The Y component represents the luminance, while the two chrominance components U and V determine the actual colour. The following expressions specify the way the RGB colour space is converted to YUV space:

$$\begin{aligned} Y &= 0.3R + 0.59G + 0.11B \\ U &= 0.493(B - Y) \\ V &= 0.877(R - Y) \end{aligned}$$

The human eye is known to be more sensitive to luminance components than to colour components [AB95]. This observation is used to separate both the luminance and chrominance parts so that the luminance component can be captured in a higher resolution than the chrominance component. Additional details on this issue can be found in [SN95].

2.2 Video Compression

Multimedia systems include media such as images, audio, video and graphics in addition to the traditional data types. Among the new media in picture, images have considerably higher storage requirements than text while audio and video need still higher data storage. The data rates required for transfer of continuous media are also significant as illustrated by the following examples [SN95]:

- For text, employing 2 bytes/character of 8×8 pixels for a screen of size 640×480 results in:

$$Characters\ per\ screen = \frac{640 \times 480}{8 \times 8} = 4800$$
$$Storage\ required\ per\ screen = 4800 \times 2bytes = 9600bytes$$
$$Storage\ required\ per\ screen \simeq 9.4kbytes$$

- An uncompressed audio signal of telephone quality, sampled at 8 kHz and quantized with 8 bits per sample would lead to:

$$Storage\ required\ per\ second = \frac{64 \times 1024\ bps}{8\ bits/byte} \times \frac{1\ sec}{1024\ bytes/kbyte}$$
$$Storage\ required\ per\ second \simeq 8\ kbytes$$

- A video sequence of 25 fps, with luminance and chrominance encoded in 3 bytes and for a frame size of 640×480 pixels requires:

$$Data\ rate = 640 \times 480 \times 25 \times 3\ bytes/sec = 23040000\ bytes/sec$$
$$Storage\ required\ for\ a\ 2\ hour\ movie =$$
$$23040000\ bytes/sec \times 60\ sec/min \times 60\ min/hour \times 2\ hour = 165.89\ Gbytes$$

These examples provide the requirements of secondary storage devices. In the case of processing uncompressed video, the requirements increase astronomically. In

addition, the transmission rates required are very high: 184.32 Mbps in the above example. Adding to the problem of data storage and data rate is the problem of relatively slow storage devices, which cannot play the multimedia data in real-time. Thus, the use of an appropriate compression technique can considerably reduce the data requirements. In the next section, we deal with various compression techniques available.

2.2.1 Classification

Video compression aims at lowering the total number of parameters required to represent the video, while maintaining perceptually good quality. These parameters are then coded either for storage or for transmission over the communication networks. The compression revolves around the different redundancies present in the video signal – spatial, temporal and psychovisual. *Spatial redundancy* occurs as the neighbouring pixels in each frame are related, exhibiting some degree of correlation. The pixels in consecutive frames are also correlated, leading to substantial *temporal redundancy.* In addition, the *human visual system* does not treat all the visual information with equal sensitivity, thereby leading to *psychovisual redundancy.* For example, the eye perceives changes to a greater extent in the luminance factor than the chrominance factor. Various compression techniques have been developed over the years taking advantage of these redundancies.

The compression techniques used in the multimedia systems can be grouped into entropy or lossless, source or lossy and hybrid techniques. Table 1 shows the different categories and their examples. *Entropy* encoding is a term used to describe compression algorithms that increase the *energy* or the information density in a message [AB95]. The input to be compressed is considered to be a simple digital sequence and the semantics of the data is ignored. This technique is used for media regardless of their specific characteristics. Entropy encoding is also called lossless as the decompression process regenerates the data completely. Run length encoding is an example of entropy coding which is used for text, images, and video or audio coding processes. Entropy encoding techniques exploit the spatial redundancy in compressing the media. For example, suppose a number occurs subsequently four times. Instead of sending the number 4 times, entropy encoding first sends the number and then its count.

Entropy Encoding	Runlength Encoding Huffman Encoding Arithmetic Encoding	
Source Encoding	Prediction	DPCM Delta Modulation
	Transformation	FFT DCT
	Layered Coding	Bit Position Subsampling Subband Coding
	Vector Quantization	
Hybrid Encoding	JPEG MPEG H.261 DVI, RTV, PLV	

Table 1: Coding and Compression Techniques: A Classification

Source encoding represents the compression techniques which compress sampled data by taking the semantics of the data into account. Source encoding techniques are lossy, since the original and encoded data stream are not identical though similar. The different source encoding techniques make use of the characteristics of a specific medium. Such characteristics can include temporal and psychovisual redundancies mentioned in the last section. For example, a content prediction technique can make use of spatial redundancies as in the case of still images [AB95].

Hybrid compression techniques are a combination of entropy and source encoding techniques. These combine well known algorithms and transformation techniques that can be applied to multimedia systems [Ste93]. For example, each hybrid technique in Table 1 uses entropy encoding in the form of run length encoding and/or a statistical compression.

Video sequences need to be compressed aggressively to reduce the data storage and bandwidth requirements. Typically, the extent of video compression to be employed is determined by the trade-off between the cost to be paid and the expected quality.

2.2.2 Some Techniques

Uncompressed video, as discussed in the beginning of this section, requires high bandwidth for transmission purposes and large storage spaces for storing. The obvious

solution is to bring down the asking rate of video. For this purpose, it is essential to have good compression techniques. In the following paragraphs, we present a brief description of some commonly used compression techniques [AB95], [11193].

1. DCT Transformation. A transformation useful in image compression is the *Discrete Cosine Transform* (DCT). This transformation converts an n×n block of elements into another block of n×n coefficients representing the two-dimensional unique spatial frequencies. The DCT function is reversible by using an Inverse Discrete Cosine Transform (IDCT) function.

The first coefficient in location (0,0) of the block represents the zero vertical and zero horizontal frequency and is called the DC coefficient. This is equal to the average value of the original elements. The other coefficients represent one or more nonzero horizontal or nonzero vertical spatial frequencies and are called AC coefficients.

The DCT and IDCT could be lossless, if the DCT encoded data are stored with perfect accuracy. In practice, however, the coefficients are stored as integers which can introduce small differences with the original data after the IDCT decoding.

2. Scalar and Vector Quantization. Quantization represents a range of values by a single value in the range. For example, converting a real number to the nearest integer is a form of quantization.

Scalar quantization is used to reduce the number of bits that are needed to store an integer. This can be achieved by dividing the integer by a quantization factor and rounding it to the nearest integer before it is stored. To retrieve the integer again, the quantized integer is multiplied by the quantization factor. This step is not lossless as the density of the domain of the integer is reduced by the quantization factor.

Vector quantization makes use of codebooks in combination with a matrix of vectors to represent an image. Instead of referring to elements directly, elements are referenced via the codebook. To transmit an image, only the references to the codebook (the vectors) have to be sent. Numerous vector quantization methods are available.

3. Entropy Coding: Huffman, Arithmetic and LZW Techniques. An *entropy* or *lossless* algorithm encodes the data based on their statistical characteristics.

Huffman algorithms assign shorter bit-patterns to characters in the message that occur more frequently and longer bit patterns to characters that occur less often.

The table which is used to find the frequency of occurrence of a character is called the Huffman-table. This table is determined before encoding is done by analyzing the statistics of the data to be encoded. For data with the same statistical characteristics the Huffman Table is incorporated into the decoder; otherwise, it has to be transmitted.

A better version of the basic Huffman algorithm is *adaptive Huffman* coding. This method outputs a bit-pattern for each character of the message, based on the occurrence of this character within the previously encoded characters; a character that occurred more frequently in the past has a smaller bit-pattern. The Huffman-table is built on the fly at both the encoder and the decoder and hence need not be transmitted.

Arithmetic coding performs better than the (adaptive) Huffman coding. This method assigns a fractional number of bits per code, instead of a fixed number of bits as in Huffman-coding. The result of an arithmetic coded message is a number between 0 and 1. This number is multiplied by the number of characters in the message. The integer part is used for decoding the next character and the fraction for decoding the rest of the message.

Lempel-Ziv-Welch (LZW) is a technique developed by Terry Welch. The best known implementation of LZW are the UNIX "compress" utility and CompuServe's Graphic Interchange Format (GIF). LZW is based on the the LZ77 and LZ78, which are dictionary based algorithms; they build up a dictionary of previously used strings of characters. The output consists of characters or references to the dictionary. A combination of a reference with a character generates a new reference in the dictionary. LZW is an improvement over LZ78. LZW uses a table of entries with an index field and a substitution-string field. This dictionary is preloaded with every possible symbol in the alphabet and thus, any symbol can be found with a reference. The encoder searches in the dictionary for the largest possible reference to the string at the input. This reference plus the first symbol of the input stream after the reference is stored in the output stream.

4. Fractal Compression. *Fractal* compression is one of the latest lossy image compression techniques. Fractals are images that recursively contain themselves. They are defined by a number of translations that include rescales, rotations and dimensional flips. The idea behind fractal compression is to automatically find a

fractal that resembles the image that must be compressed. A major advantage of this technique is the ability to decompress to any given resolution. This technique is unattractive for real-time image compression due to higher search time to find the transformations which can best represent a particular image.

5. Wavelet Compression. *Wavelet* transformation is a relatively new and promising development in the area of lossy image compression. An important characteristic of this transformation is that, if it is applied on a time-domain signal, it results in a representation that is localized in the time domain as well as in the frequency domain.

The wavelet transformation converts a sample of 2^J values into 2^{J-1} approximate wavelet transform coefficients and 2^{J-1} detail wavelet transform coefficients. This transformation can be repeated over the generated approximation wavelet transform coefficients a number of times, until the minimum number of two approximate wavelet transform coefficients and $2^J - 2$ detailed transform coefficients remain. The number of transformations is called the number of *levels* of the wavelet transformation. The wavelet transformation is inversive. Thus, by applying the inverse wavelet transform equal to the number of levels, the original sample can be recomputed.

2.3 Video Compression Standards

The development of digital video compression techniques has made many telecommunication applications feasible. These include teleconferencing and video telephony. Standardizing video compression techniques was necessary to bring down the cost of the codecs and to resolve the problem of interoperability of equipments from different vendors. The development of the MPEG standard derived technical input from earlier standardizing bodies like JPEG for still images and the CCITT[1] group on visual telephony. It is thus essential to know the developments of these standardization bodies before presenting the MPEG standard. Hence, in the following subsections we present three MPEG related standards.

[1]CCITT is the International Committee on Telegraph and Telephones.

2.3.1 The Joint Pictures Expert Group

Joint Pictures Expert Group (JPEG) played a considerable role in the beginning of the MPEG as both were in the same working group of the ISO [Ste93]. In JPEG, the picture is divided into blocks of 8×8 pixels for encoding. Each block is transformed into another block of 8×8 using the DCT function to obtain 64 unique two-dimensional spatial frequency coefficients. Next, the coefficients are quantized using an 8×8 quantization table. In this step, each coefficient is divided by its corresponding quantizational value with the result rounded to the nearest integer. The final step involves entropy encoding consisting of a mixture of the variable-length encoder and the Huffman or arithmetic encoder.

Despite the fact that the JPEG focussed exclusively on still image compression, the distinction between still and moving image is thin; a video sequence can be thought of as a sequence of still images to be coded individually, but displayed sequentially at video rate. It is to be noted that in this case "the sequence of still images" fails to take into account the interframe redundancy present in the video sequences. Hence, in order to exploit the temporal redundancy which could bring down the storage and bandwidth requirements considerably, extending the ISO committee to moving pictures was a natural step.

2.3.2 CCITT Expert Group on Visual Telephony

The high growth oriented nature of the video compression area was triggered by applications such as teleconferencing and video-telephony [Gal91]. The definition and planned deployment of ISDN was the motivation for the development of compression techniques at the rate of p×64 kbps, where p takes values from 1 to 30. The visual telephony group produced the recommendation H.261: "*Video Codec for Audiovisual Services at p×64 kbits*". The focus of H.261 is the real-time encoding-decoding system, which exhibits less than 150 ms delay.

H.261 has two modes of coding: *intraframe* and *interframe* [AB95]. The intraframe encoding is similar to JPEG, where each block of 8×8 pixels is DCT transformed. The quantization of DC-coefficients differs from that of AC-coefficients. In the last step, the AC and DC parameters are entropically encoded to produce a variable length word. In interframe mode, each 8×8 block is also DCT transformed and system quantized, but the result is first sent to the motion-compensator and then to the

entropy encoder. The motion-compensator is used for comparing the macroblock of the current frame with blocks of the previously sent frame. If the difference is below a predetermined threshold, no data are sent for this block. Otherwise, the difference is DCT transformed and linearly quantized. In the final step, a variable length stream is obtained from the entropy encoder.

The MPEG committee on pursual of the CCITT expert group's work concluded that relaxing the constraint on very low delay and focusing on extremely low bit rates can lead to a solution with increased visual quality in the range of 1 to 1.5 Mbps. Thus, the decks were cleared for the development of MPEG-1.

2.3.3 CMTT/2 Activities

The use of digital video compression for video-conferencing and video-telephony also opens up the issue of its use in broadcasting compressed television signals [Gal91]. The transmission channels for such an application are either the high level digital hierarchy-H21 (34 Mbps) and H22 (45 Mbps) or digital satellite channels. The CMTT/2[2] addressed the compression of television signals at 34 and 45 Mbps. The focus of this work was to produce quality codecs. While the technology used might have some commonalities with MPEG, the problem and target bandwidth are very different.

2.4 Motion Pictures Expert Group

Amidst the largely uncorrelated independent initiatives described in the previous section, the Motion Pictures Expert Group (MPEG) was established. The two main challenges addressed were: first, to find a mechanism to convince the different industries that there was a technology advantage in going digital, together with the same solution and second, to define a single "syntax" capable of representing the audiovisual information in such a way that the common syntax could be the common platform enabling interoperability between applications.

The MPEG-1, part of the ISO-IEC/JCT1/SC2/WK11, was entrusted with the development of an audio-visual standard [Gal91], [Chi95], [Fur94]. The premise of

[2]CMTT is a joint commission of the CCITT and the CCIR – the International Consultative Committee on Broadcasting.

MPEG-1 is that the video signal and associated audio can be compressed to a bit rate of 1.5 Mbps. This could well be the starting point where the video can be a form of computer data type. The MPEG-1 is the first standard of the MPEG-Group and consists of three parts: MPEG-1-Video, MPEG-1-Audio and MPEG-1-Systems. MPEG-1-Video addresses the compression of digital video at about 1.5 Mbps. MPEG-1-Audio addresses the compression of digital audio signal at the rates of 64, 128 and 192 kbps per channel. MPEG-1-System addresses the issue of synchronization and multiplexing of multiple compressed audio and video bit streams. Since the thrust of our work is video, we focus on the MPEG-1-Video in the rest of this section.

2.4.1 Bitstream Details

The syntax for the MPEG-1-Video sequence can be specified in the BNF notation as follows [LCY94]:

$$
\begin{aligned}
\langle sequence \rangle &::= \langle sequence\ header \rangle \langle group\ of\ pictures \rangle \\
&\quad \{[\langle sequence\ header \rangle] \langle group\ of\ pictures \rangle\} \\
&\quad \langle sequence\ end\ code \rangle \\
\langle group\ of\ pictures \rangle &::= \langle group\ header \rangle \langle picture \rangle \{\langle picture \rangle\} \\
\langle picture \rangle &::= \langle picture\ header \rangle \langle slice \rangle \{\langle slice \rangle\} \\
\langle slice \rangle &::= \langle slice\ header \rangle \langle macroblock \rangle \{\langle macroblock \rangle\}
\end{aligned}
$$

where the curly brackets {} delimit an expression that is repeated zero or more times.

The *sequence header* contains control information needed to decode the MPEG-1 video sequence [11193]. It starts with a unique sequence header code. The horizontal and vertical sizes determine the size of the picture. The picture rate field gives the frame display rate and there are 8 different rates to choose from, in the set of {23.976, 24, 35, 19.97, 30, 50, 59.94, 60} fps. The additional fields include: pel aspect ratio, VBV buffer size, extension data, user data and other flags.

The pictures are united into groups to facilitate random access. The time code included in the *Group of Picture* header facilitates decoding at intermediate points.

Each *picture* has a header which contains, among other items, the temporal reference and the picture type. The temporal reference is used to define the order in

which the pictures must be displayed. The picture type distinguishes between the four types of pictures allowed in the MPEG-1, they are:

- *Intra coded frames* (I-frames) are self contained. In other words, they are coded without any reference to other images. The compression of I-frames is the lowest within MPEG. I-frames serve as points for random access.

- *Predictive coded frames* (P-frames) require information of the preceding I or P-frame for encoding and decoding. The achievable compression ratio of P-frames is considerably higher than that of I-frames.

- *Bidirectionally predicted frames* (B-frames) require, for encoding and decoding, information of the preceding and following I and/or P-frames. B-frame compression achieves the highest compression ratio. A B-frame is defined as the difference of a prediction of the past image and the following P or I-frame. It can never be accessed in a random fashion.

- *DC coded frames* (D-frames) are intra frame encoded. They can be used in fast forward or rewind operations. The DC-parameters are DCT-coded and the AC-coefficients are neglected.

Each picture is divided into *slices*. Each slice consists of an integral number of macroblocks in the raster scan order. There can be one or more slices in a picture and their size can vary. Each slice starts with a slice header code, which defines the vertical position of the slice. This is followed by the quantization step-size. The horizontal start position of the slice is given by the macroblock address of the first macroblock in the slice. The result of all this is that, within a picture, a slice can be decoded without information from the previous slices. Therefore, if an error occurs, decoding can begin again at the subsequent slice.

Slices are divided into *macroblocks* of 16×16 pixels. The macroblock header contains information about the macroblock address, macroblock type and optional quantizer scale.

A *block* is an array of 8×8 component pixel values treated as a unit and input to the DCT. Blocks of 8×8 pixels are transformed into an array of 8×8 DCT coefficients using the two dimensional DCT.

Sequence Layer:	(Random Access Unit: Context)
Group of Pictures Layer:	(Random Access Unit : Video Coding)
Picture Layer:	(Primary Coding Unit)
Slice Layer:	(Resynchronization Unit)
Macroblock Layer:	(Motion Compensation Unit)
Block Layer:	(DCT Unit)

Table 2: Six Layers of MPEG-1 Video and Their Functions

Each of the above six layers supports a definite function, which can be either a signal processing function (DCT or motion compensation) or a logical function (resynchronization or random access point). The layers and their functions are summarized in Table 2 [Gal91].

2.4.2 MPEG-1 Compression Algorithm

The main aspect which stands out between a still image compression technique like JPEG and a moving picture compression technique like MPEG, is *interframe coding.* This aspect stems out, since the quality requirements demand a very high compression not achievable with intraframe coding alone. On the other hand, the random access requirement is best met with intraframe coding. Addressing all these aspects requires a delicate balance between intraframe and interframe coding, and between recursive and non recursive temporal redundancy reduction. This challenge is answered by the MPEG-Group by using two interframe coding techniques: *predictive* and *interpolative.*

The MPEG-1 video compression algorithm relies on two basic techniques: block based motion compensation for reduction of the *temporal redundancy* and transform based (DCT) compression for the reduction of the *spatial redundancy* [Gal91], [11193]. Motion compensation techniques are applied with predictors that are both casual (pure predictive coding) and non-casual (interpolative coding). The remaining signal (prediction error) is further compressed with spatial redundancy reduction (DCT). The information relative to motion is based on 16×16 pixel blocks and is transmitted together with the spatial information. The motion information is compressed using variable-length codes to achieve maximum efficiency.

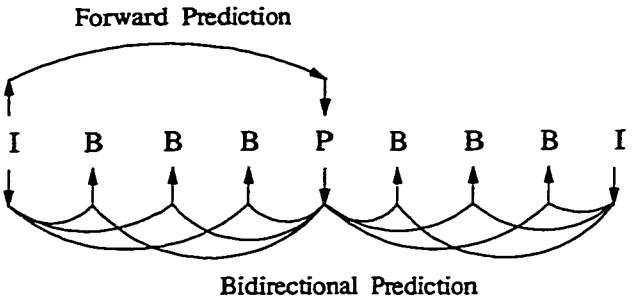


Figure 1: Dependency Among the I, P and B Frames

Temporal Redundancy Reduction. The three types of frames: I, P and B address the dual need of random access points and significant reduction in bit-rate. In case of P and B frames, which are coded with respect to a reference frame, motion compensation is used to improve the coding efficiency. The relationship between the three types of frames is illustrated in Figure 1.

The organization of the pictures in a sequence depends on the application specific parameters such as random accessibility and coding delay.

Motion Compensation. Motion-compensated *prediction* is the most widely used temporal redundancy reduction mechanism. Motion-compensated prediction assumes that "locally" the current picture can be modelled as a translation of the picture at some previous time. Locally means that the displacement need not be the same everywhere in the picture. In this process, motion information is a crucial part to recover the picture.

Motion-compensated *interpolation*, also called the bidirectional prediction, is the key feature of MPEG. In the temporal dimension, Motion-compensated interpolation is a multi-resolution technique: a sub-signal with low temporal resolution (typically 1/2 or 1/3 the frame rate) is coded and the full resolution signal is obtained by interpolation of the low-resolution signal and addition of a correction term. The signal to be reconstructed by interpolation is obtained by adding a correction term to a combination of a past and a future reference.

Motion Representation: Macroblock. The choice of 16×16 pixels blocks for motion compensation is a result of the tradeoff between the coding gain provided by the motion information and the cost associated with coding the motion information.

These motion compensation blocks are called *macroblocks*. In the case of a bidirectionally predicted picture, each 16×16 macroblock can be of type intra, forward-predicted, backward-predicted or average. The motion information consists of one vector each for forward-predicted macroblocks and backward-predicted macroblocks, and two for bidirectionally predicted macroblocks. The motion information associated with each 16×16 block is coded differentially with respect to the motion information present in the adjacent block. The range of the differential motion vector can be selected on a picture-by-picture basis to match the spatial resolution, the temporal resolution and the nature of the motion in a particular sequence. The differential motion information is further coded by means of a variable-length code to provide greater efficiency by taking advantage of the strong spatial correlation of the motion vector field.

Motion Estimation. Motion estimation covers a set of techniques used to extract the motion information from a video sequence. The MPEG syntax specifies how to represent the motion information: one or two motion vectors per 16×16 sub-block of the picture depending on the type of motion compensation used. The MPEG draft does not specify how such vectors are to be computed. However, block-matching techniques are the likely choices as the motion representation is block-based. In a block-matching technique, the motion vector is obtained by minimizing a cost function that measures the mismatch between a block and each predictor candidate.

Spatial Redundancy Reduction. Both still-image and the prediction error signal have a very high spatial redundancy. There are many redundancy reduction techniques usable to this effect, but due to the block based nature of the motion compensation process, block based techniques are preferred. In this context, transform coding and vector quantization techniques are two likely candidates. Transform coding techniques with combination of visually weighted scalar quantization and run length coding have been preferred because the DCT provides a definite number of advantages and has a relatively straight forward implementation.

Discrete Cosine Transform. The DCT has inputs in the range [-255, 255] and output signals in the range [-2048, 2048], providing enough accuracy even for the finest quantizer. In order to control the effect of rounding errors when different implementations of the inverse transforms are in use, the accuracy of the inverse

transform is determined according to the CCITT H.261 standard specification.

Quantization. Quantization of the DCT coefficients is a key operation, because the combination of quantization and run length coding contributes to most of the compression. It is also through quantization that the encoder can match its output to a given bit rate. Additionally, the adaptive quantization is the key to achieving visual quality. As the MPEG has both intra coded and differentially coded pictures, it combines features of both the JPEG and H.261 standards to achieve a set of accurate quantization tools to deal with the DCT-coefficients. In the following paragraphs, we present different forms of quantizations employed in video compression.

1. Visually Weighted Quantization. Subjective perception of quantization error varies with the frequency thus, it is advantageous to use coarser quantizers for the higher frequencies. The exact quantization matrix depends on many external parameters such as the characteristics of the intended display, the viewing distance and the amount of noise in the source. Hence, it is possible to design a particular quantization matrix for an application or an individual sequence which can be stored as context along with the compressed video.

2. Quantization of Intra versus Non-intra Blocks. The signal from intra coded blocks is quantized differently from the signal that results from prediction or interpolation. Intra coded blocks contain energy in all frequencies and are very likely to produce "blocking effects" if too coarsely quantized. On the other hand, prediction error-type blocks contain predominantly high frequencies and can be subjected to much coarser quantization.

3. Modified Quantizers. The human visual system does not perceive all spatial information alike and some blocks need to be coded more accurately than others. This is particularly true of blocks that correspond to very smooth gradients where a very slight inaccuracy could be perceived as a visible block boundary (blocking effect). Such inequality between blocks can be dealt with by modifying the quantizer step size on a block-by-block basis if the image content makes it necessary. Furthermore, this mechanism can also be used to provide a very smooth adaption to a particular bit rate.

Entropy Encoding. In a further measure to improve the compression inherently present in the DCT and reduce the impact of motion information on the total bit rate, variable-length coding is used. Only those codes with a relatively high probability of occurrence are coded with a variable-length code. The less-likely events are coded with an escape symbol followed by a fixed length code to avoid extremely long code words thus reducing the cost of implementation.

2.4.3 MPEG-2 and MPEG-4

MPEG-1 targeted applications at about 1.5 Mbps. In due course, attention turned towards higher rate applications, and the result was MPEG-2. This mainly targets broadcast video between 4 to 10 Mbps [AB95]. It is efficient for other applications with higher bit rates and sample rates like HDTV. Interlaced video streams which are common in broadcast television find MPEG-2 to be more suitable than MPEG-1. MPEG-2 handles different video streams which are divided into profiles and levels.

MPEG-4, on the other hand is not yet an international standard. It addresses low bit rate coding from 8 to 64 kbps. MPEG-4 organizes the scene as a composition of arbitrarily shaped discrete pieces rather than rectangles of digitized data samples with corresponding audio [KPC97]. This is slated to become a standard by 1998.

Chapter 3

Networks

Communication networks offer a variety of services ranging from the basic telephone service to current multimedia based integrated services. These services show distinctly different characteristics in terms of the scope of users and applications, and the integration of different applications and media. Thus, this chapter first captures the network requirements of the emerging multimedia applications. We then present the Asynchronous Transfer Mode networking technology, which supports real-time multimedia traffic. As majority of the traffic flows through the Internet, we next describe how the Internet supports multimedia applications in association with advanced protocols. Since our work concerns link and associated buffer sharing, we further look at different ways of traffic scheduling. We then present how ATM can support TCP/IP as most networks connected to the Internet run TCP/IP. Finally, we discuss how quality of service can be met for video, the medium we are studying.

3.1 Networking Requirements for Multimedia Applications

Multimedia communication imposes many requirements on the network used. In general, these requirements are determined based on the respective application. Media, such as video, are characterized by large frame sizes, calling for much higher resources than traditional media such as text. The different media in multimedia applications such as audio and video are bounded by processing time. Such bounds impose stringent constraints on the delay applications can tolerate. Delays beyond

the expected levels can render a segment of data useless, as in the case of a video clip where the associated audio has to be played at a particular time instant. Multimedia applications demand measures against data corruption or data loss or both. If data are missing or corrupt, there need to be mechanisms to recover from such impairment. Furthermore, it is natural for multimedia applications to employ more than two media to convey the information. Thus, it becomes necessary to maintain their relationship with respect to time. All these features become necessary to maintain the quality at the receiver's end. In the following paragraphs, we present details about the requirements of multimedia on network technologies [Kar95], [Dav92].

Bandwidth. Among the different media, transmission of compressed video requires one of the highest data rates, and accordingly, network experts predicted that the requested rate of such media would drive the bandwidth requirements in the 1990s [Stu95]. In order to reduce the transmission bandwidth and storage requirements, video is compressed in today's applications. We look at three video standards: MPEG, DVI and H.261. These standards require about 1.5 Mbps for MPEG-1, 4 Mbps and above for MPEG-2, 1.2 to 1.8 Mbps for DVI and 0.064 to 2 Mbps for H.261. Practical experience recommends a total of 1.4 Mbps for audio and video for DVI and MPEG-1 [Stu95]. The proposed bandwidth provides good video quality and supports commercial audiovisual equipment. Additionally, it can be transmitted over the T1 leased lines of capacity 1.54 Mbps [Bat92]. Apart from this bandwidth capacity, the highly asymmetric nature of the traffic requires attention in dealing with bandwidth management.

Transmission Delay. The delay restrictions of multimedia applications are more stringent than those of traditional data traffic. In order to show the difference in magnitude (but not actual values), consider the example of traditional traffic, which consists of a file transfer and video. The file transfer can have a maximum end-to-end delay of 1 sec, while video can have 0.25 sec [HSS90]. Consider another example, voice communication via satellite with round-trip transmission time of around 0.6 second makes it very difficult for partners to hold a normal conversation. In this case, an end-to-end delay of 0.3 second is desired. Practical experience suggests an end-to-end delay of 150 ms for interactive video applications [Stu95].

Based on the delay characteristic, traffic can fall into one of the *three* categories

[SN95]. First, the *asynchronous transmission mode* directs communication with no restrictions on time. Second, the *synchronous transmission mode* defines a bounded transmission delay for each message. Third, the *isochronous transmission mode* defines a constant transmission delay for each message.

Audio and video fall into the isochronous category. Isochrony does not have to be maintained across the entire path from source to sink, only at the information's final destination. Thus, isochrony can be recovered by a *play-out* buffer at the sink. The cost to be paid for this is the additional delay due to the play-out buffer [Stu95]. For practical purposes, synchronous traffic with sufficiently low delay bound can fake an isochronous stream between source and sink. Therefore, the delay variance or jitter is not considered as a separate item. To meet CCITT requirements, the maximum delay bound is assumed to be 150 ms. We can break this into at least four components:

1. source compression and packetization delay,
2. transmission delay,
3. end-system queueing and play-out (synchronization) delay,
4. sink decompression, depacketization and output delay.

Reliability. Existing communications systems strive to provide end-to-end reliability by using checksum and sequence numbering for *error control* and some form of negative or missing positive acknowledgement with packet retransmission handshake for *error recovery* [SN95]. The system performance will be very bad if checksumming is not performed either in hardware or at the media access layer or link layer, as it consumes more time thereby reducing the performance [Kes97]. Error correction through retransmission due to a negative acknowledgement is not appropriate for time critical data as the retransmitted data would normally arrive late.

A possible remedy for the conflicting goals of reliability and low delay is to use *Forward Error Correction* (FEC) techniques. FEC takes a set of input symbols representing data and adds redundancy to produce a different and larger set of output symbols. It is advantageous to employ FEC technique as the receiver can recover from a packet loss without retransmission. For time bound data like audio and video, the retransmission delay, which at the minimum can be of the range of one round trip time, is not favourable. As a result, FEC schemes are suitable for real-time traffic,

especially for links with long propagation delays. There are *disadvantages* with FEC [Kes97]. First, the load from a source may increase due to error correction. As packet losses increase with load, FEC may tend to cause more losses, degrading overall performance. Second, FEC is not effective when the packet losses are bursty, a case similar to that of high-speed networks. Finally, FEC leads to an increase in end-to-end transmission time, as the receiver has to wait for the entire FEC block before processing the packets in the block. On the other hand, if the data from a stream is spaced well enough not to be affected by bursty losses and their error correction overhead is small then FEC may perform well.

Synchronization. An important feature of multimedia systems is *integrated* media processing. The need for integration arises because of the inherent dependencies among the information coded in media objects. Though the word *synchronization* refers to time, in multimedia it is often used to reflect content, spatial, and temporal relationships between media objects [SN95]. For instance, when audio, video, and other data streams come from different sources via different routers, there is a need for mechanisms to synchronize these different streams at the destination in order to achieve the equivalent of lip synchronization. Synchronization can be achieved using a combination of time-stamping and play-out buffers [Stu95].

3.2 The Asynchronous Transfer Mode

In an effort to provide real-time transport capability for emerging multimedia applications, networks with high bandwidth and low latency are required. For an application aiming to transmit large amounts of data in real-time, new network architectures and protocols need to be designed. Furthermore, such architectures and protocols must support the different media types in an efficient and cost effective way. Thus, the development of the *Asynchronous Transfer Mode* (ATM) was initiated. In 1988, CCITT (now ITU) designated ATM, a cell based switching technology, as the transmission mode for the future broadband ISDN services [For95], [Sat96], [SJ95], [Vet95], [Ste95]. At the same time the ATM Forum, driven by more than 700 members, including almost all major providers of data and communication equipment, focused on the definition of the private ATM networks for the local and wide area environments.

ATM networks are designed for high bandwidth, scalability and manageability.

ATM uses statistical multiplexing to provide *bandwidth on demand*, where the clients negotiate bandwidth allocation rather than the choice being made by the network. ATM is potentially capable of supporting all classes of traffic (audio, video and traditional), in one transmission and switching fabric technology. It promises to provide greater integration of capabilities and services, and increased and more flexible access to the network.

3.2.1 Basic Principles

ATM networks carry all traffic on a stream of fixed-size packets called *cells*, each comprised of 5 bytes of header information and a 48-byte information field, which is often referred to as *payload*. The cell size was chosen to be fixed to enable quick switching and multiplexing. ATM is connection-oriented technology. In ATM networks, each connection is called a *virtual circuit* or *virtual channel* (VC), because it also allows the capacity of each link to be shared by connections using that link on a demand basis rather than by fixed allocations. The connections allow the network to guarantee the *quality of service* (QoS) by limiting the number of VCs, where QoS determines the end-to-end network performance through parameters mentioned later in subsection 3.2.3. Typically a user declares key service requirements at the time of connection setup. In addition, they declare the traffic parameters and may agree to control these parameters *dynamically* as demanded by the network.

3.2.2 ATM Protocol Reference Model

The ATM protocol reference model is based on standards developed by the ITU. Communication from the higher layers is adapted to the lower ATM defined layers, which in turn pass the information onto the physical layer for transmission over a selected medium. The protocol reference model consists of three layers: the *ATM adaptation layer*, the *ATM layer*, and the *Physical layer* as given in Figure 2. The reference model contains three planes; a *user plane* to transport user information, a *control plane* mainly composed of signalling information and a *management plane* to maintain the network and to perform operational functions. In the rest of this section we present the three layers in detail.

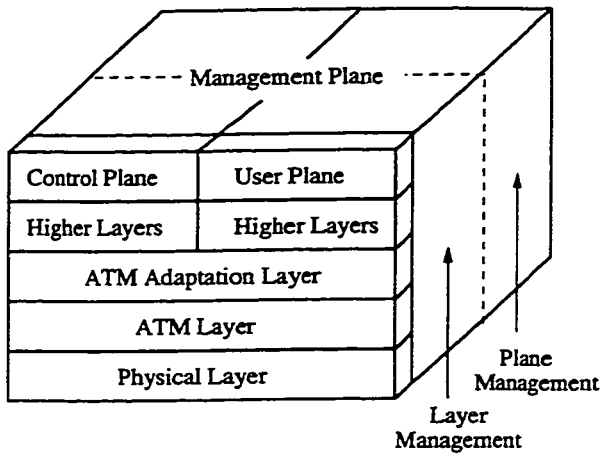


Figure 2: ATM Protocol Reference Model

1. The ATM Adaptation Layer (AAL). The AAL interfaces the higher layer protocols to the ATM layers. It *relays* ATM cells both from the upper layers to the ATM layer and vice versa. When relaying information received from the higher layers to the ATM layer, the AAL segments the data into ATM cells. When relaying information received from the ATM layers to the higher layers, the AAL must take the cells and reassemble the payloads into a format the higher layers can understand. This is called *segmentation and reassembly.* To support different types of traffic or service, four types of AALs were initially proposed [Vet95]. In the recent ATM Forum Traffic Management Specification Version 4.0, five service categories are specified. While each of the AALs is optimized for a particular type of traffic, there is no stipulation in the standard that AALs designated for one service cannot be used for another. In the following paragraphs we present details of different service classes [Sat96].

Constant Bit Rate service (CBR). AAL1 supports a connection-oriented service in which the bit rate is constant. Examples of such applications include 64 kbps voice, fixed rate uncompressed video and leased line for private data networks. Under CBR, the network application establishes a CBR connection and negotiates the parameter called *Peak Cell Rate* (PCR), the maximum data rate the connection will support without losing data.

Variable Bit Rate service (VBR). AAL2 supports a connection oriented service in which the bit rate is variable but requires a bounded delay for delivery. Examples for this service include compressed packetized voice and video.

Real-Time VBR service (RT-VBR). This category is meant for real-time applications requiring tightly constrained delay and jitter as in the case of live video and audio. In addition to PCR, RT-VBR is characterized by *Sustained Cell Rate* (SCR), the average throughput rate the application is permitted and the *Maximum Burst Size* (MBS), the maximum number of back-to-back cells that can be sent at the PCR. For this category, the sources are expected to transmit at a rate which varies with time or, in other words, the sources can be described as *bursty.*

Non-Real-Time VBR service (NRT-VBR). This category is intended for non-real-time applications which have bursty traffic characteristics. This category is characterized in terms of PCR, SCR and MBS, but no delay bounds are associated with this service category. For the cells transmitted in this category, the application expects a low cell loss. In addition, this category may support statistical multiplexing.

Available Bit Rate service (ABR). This service category permits the parameters representing ATM layer transfer characteristics, which include PCR and *Minimum Cell Rate* (MCR), to change subsequent to connection establishment. A flow control mechanism is used to control the source rate. Feedback is provided to the source through the control cells called *resource management cells.* It is expected that an end-system that adapts its traffic in accordance with the feedback will receive low cell loss and get a fair share of the bandwidth. Furthermore, ABR service does not need to bind the delay or the delay variation experienced by a connection.

Unspecified Bit Rate (UBR). This class is intended for non-real-time applications that do not need tightly bound delay and jitter. Examples for this category include traditional computer communication applications such as file transfer and e-mail. The UBR service does not specify bit-rate and quality of service guarantees. The only assurance is that UBR will make its best effort to deliver cells. There is no flow-control. Thus, the cells are subject to drops once the buffers are full.

2. The ATM Layer. The ATM layer provides an interface between the AAL and the physical layer. The ATM layer is responsible for [SN95], [For95]:

- relaying cells from the AAL to the physical layer for transmission and from the physical layer to the AAL for use at the end system;
- translation of the cell identifier and management functions;

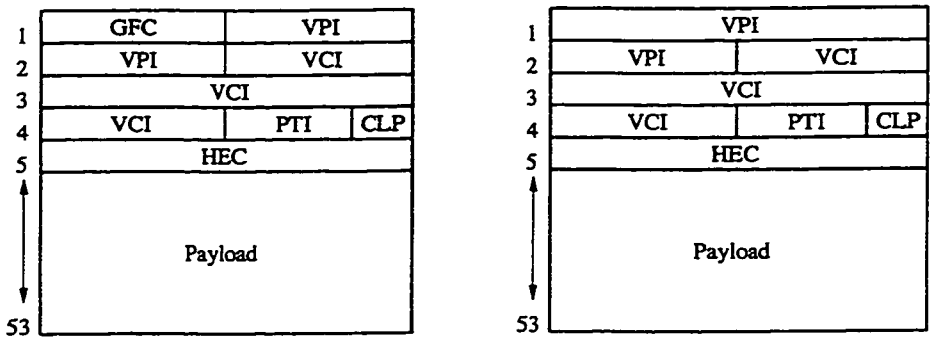


Figure 3: UNI (left) and NNI (right) ATM Cell Format

- multiplexing and demultiplexing of cells of different connections into a single cell stream;
- cell header insertion, which includes explicit forward congestion and payload indicators;
- implementation of a generic flow control mechanism at the user-network interface;
- monitoring the transmission rate and enforcing the traffic contract.

The ATM layer's functionality is defined by the fields in the ATM cell header. The format of the header for ATM cells has two different forms; one for use at the *user-to-network interface* (UNI) and the other for internal use by the network, the *network-to-network interface* (NNI).

As shown at the beginning of Figure 3, four bits are allocated to *Generic Flow Control* (GFC), which is used by the UNI to control the amount of traffic entering the network. This allows the UNI to limit the amount of data entering the network during periods of congestion. At the NNI these four bits are assigned to the *Virtual path Identifier* (VPI).

The VPI and the *Virtual Channel Identifier* (VCI) together form the routing field, which associates each cell with a particular channel or circuit. The VCI is a single channel identifier; the VPI allows grouping of VCs with different VCIs and allows the group to be switched together as an entity.

The *Payload Type Indicator* (PTI) field is used to distinguish between user cells and control cells. This allows control and signalling data to be transmitted on a different sub-channel from user data.

The *Cell Loss Priority* (CLP) provides the network with selective discard capability. This bit can be set by the user to indicate lower priority cells that could be discarded by the network during periods of congestion. For example, voice data may be able to suffer lost cells without the need for retransmission, whereas text data cannot tolerate such losses. The CLP can also be used by the network to indicate cells that exceed the negotiated rate limit of the user; a process called *tagging.*

The *Head Error Check* (HEC) field is used to reduce errors in the header that cause a misrouting of the cell for one user into another user's data stream. This field contains the result of an 8-bit CRC checking on the ATM header.

3. The Physical Layer. The physical layer encodes and decodes the data into suitable *electrical/optical* waveforms for transmission and reception on the communication medium used. In addition, the physical layer also provides cell delineation functions, HEC generation and processing, performance monitoring and payload matching of the different formats used at this layer.

The physical layer can *transfer* ATM cells from one user to another in two ways. At the UNI, ATM cells may be carried in an externally framed synchronous structure or in a cell-based asynchronous transmission structure. The SONET (Synchronous Optical NETwork), a synchronous transmission structure, is often used for framing and synchronization at the physical layer.

3.2.3 Traffic Contract

The ATM network, in order to deliver guaranteed service on demand, employs many *traffic management mechanisms.* At the same time, the utilization of available network resources has to be maximized. The first step in this process is the specification of the traffic contract. A user requesting a connection with the ATM network has to declare the characteristics of the connection in terms of the traffic contract parameters. We enumerate the set of traffic contract parameters along with their definitions below [SJ95]:

1. *Peak Cell Rate* is defined as the static amount of bandwidth that is continuously available or the maximum instantaneous rate at which the user will transmit.

2. *Sustained Cell Rate* is defined as the average rate as measured over a long time interval.

3. *Cell Loss Ratio* is defined as the percentage of cells lost in the network due to error or congestion and not delivered to the destination.

4. *Cell Transfer Delay* is defined as the delay experienced by a cell between network entry and exit points. This includes the propagation delay, queueing delay at various intermediate switches and service times at queueing points.

5. *Cell Delay Variation* is defined as the variance of cell transfer delay. High variation implies larger buffering for delay sensitive traffic such as voice and video.

6. *Burst Tolerance* (BT) is defined as the maximum burst size (MBS) that can be sent at the PCR. This is the bucket size parameter for the leaky bucket algorithm employed to control the traffic entering the network. BT and MBS are related as follows [SJ95]:

$$BT = (MBS - 1)\left(\frac{1}{SCR} - \frac{1}{PCR}\right)$$

7. *Minimum Cell Rate* is defined as the minimum cell rate desired by the user.

3.2.4 Traffic Management

The ATM traffic management specification defines a set of traffic and congestion control functions to monitor the network. These functions are intended to react to network congestion in order to minimize its intensity, spread and duration. They are as follows [Sat96]:

- *Connection Admission Control* is the set of actions taken by the network during connection establishment to determine whether the connection can be accepted or rejected.

- *Usage Parameter Control* is the set of actions taken by the network to monitor and control traffic. Its main purpose is to protect the resources from violations of the negotiated traffic contract, which can affect the QoS of other already established connections.

- *Selective Cell Discard* is an optional function. A network under congestion can selectively discard cells which meet either of the following conditions:

1. cells that belong to a non-compliant ATM connection, where a non-compliant connection is defined as the one whose cells do not conform with the traffic contract;
2. cells that have the CLP bit in the cell header set to 1.

- *Traffic Shaping* is a mechanism that alters the characteristics of a stream of cells on a connection to achieve better network efficiency while meeting the QoS objectives or to ensure conformance at a subsequent interface.

- *Explicit Forward Congestion Indication* (EFCI) is also an optional function. A network element in an impending congested state or a congested state may set an EFCI in the cell header so that this field can be examined by the destination end-system to take some corrective action.

- *Resource Management* uses Virtual Paths, which are an important component in traffic control and resource management in ATM networks. Virtual Path connections can be used in conjunction with traffic control to:

  1. simplify call admission control;
  2. implement a form of priority control by segregating groups of virtual connections according to service category;
  3. efficiently distribute messages for the operation of traffic control schemes;
  4. aggregate user-to-user services such that the UPC can be applied to the traffic aggregate.

- *Frame Discard* is based on the premise that, in case a network needs to discard cells, it is more effective to discard at the frame level rather than at the cell level [RRV93]. The frame referred in this case is the AAL protocol data unit. The network detects frame boundaries by checking the payload type field in the cell header. This mechanism can help avoid congestion and increase the performance by preventing partial data from consuming bandwidth [Sat96]. For example, in the case of video, instead of sending only a few packets belonging to a frame rendering the frame incomplete, it is better not to send such incomplete frames which may eventually be dropped.

- *ABR Flow Control* needs the transmitting stations on an ATM network to increase or decrease their transmission rates according to the network's available bandwidth, reducing congestion and cell loss. Transmitting stations collect bandwidth information by generating resource management cells. The resource management cell collects bandwidth information along its way to its destination node, which returns bandwidth data to its transmitting source so that the subsequent transmissions can be optimized.

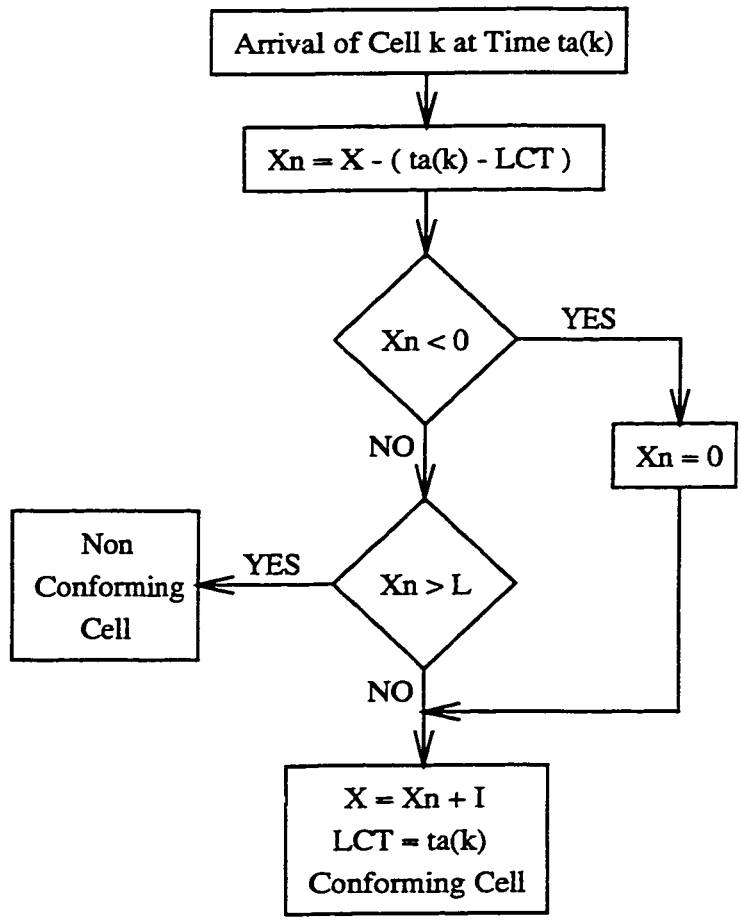
3.2.5 Traffic Management Related Algorithm

The *Generic Cell Rate Algorithm* (GCRA), also called the *leaky bucket*, is used to define conformance in accordance with the traffic contract. The Usage Parameter Control function may implement the GCRA, or one or more equivalent algorithms, to enforce conformance. Although traffic conformance is defined in terms of GCRA, the network is not required to use this algorithm for UPC. The continuous state leaky bucket [For95], a version of GCRA, is defined by the flow chart in Figure 4.

3.3 Internet

The Internet is a loose collection of networks organized into multi-level hierarchies using a wide variety of interconnection technologies. It is often referred to as a network of networks. The majority of traffic on the Internet consists of applications that are insensitive to the performance they receive from the network. For example, consider a file transfer application. The best possible scenario is infinite bandwidth and zero end-to-end delay. However, this application works fine as the available bandwidth decreases and delay increases. Such applications are termed *elastic*, as they can adapt to changing situations. These applications are called *best effort service* applications. But, presently there are applications which need a bound on the end-to-end delay. For example, an audio application carrying voice as a 64 kbps requires the round trip delay to be around 150 ms [Kes97]. Applications like this, which need a guarantee from the network on the performance, are called *guaranteed service* applications. For guaranteed delay bound applications, any packets that arrive at the sink beyond the prescribed delay will be discarded thereby degrading the performance.

The emerging multimedia applications have a different requirement to achieve



X	Value of the Leaky Bucket Counter
Xn	Auxilary Variable
LCT	Last Compliance Time
I	Increment
L	Limit

At the time of arrival ta of the first cell
of the connection, X = 0 and LCT = ta(k)

Figure 4: Generic Cell Rate Algorithm

desired performance and, at the same time, have different data characteristics. These are time critical applications, which are less elastic and less tolerant of delay variation. If such applications were to be run on the current Internet framework of best effort service, they will be subjected to excessive delay and, in turn, a large number of packet drops. In addition, these applications do not adapt to congestion and this leaves less bandwidth for other adaptable applications. It is thus necessary to extend the current Internet model to support the newer applications [She95]. The IETF's *Internet Services Group* is working on providing an enhanced model.

3.3.1 Integrated Services

As any other network, the goal of *Extended Internet* is to maximize performance. Extended Internet has to support applications that do not fit into the best effort category. In order to support such applications, there must be a mechanism to meet the performance requirements of the emerging applications. These requirements can be met by providing multiple service classes instead of the prevailing best effort service class. The new set of service classes should reflect the concerned application's requirements. It is also essential to support any application in the prescribed multi-service class. Thus, instead of a network implicitly choosing the service class for a particular application it would be better for it to be explicitly specified by the application. Additionally, admission control is another feature that needs to be supported by the network. Such a mechanism would allow the network to reject incoming requests to preserve the performance levels of applications already being serviced. Also, this will provide a way for the applications to specify their requirements, based on which, the network can either accept or reject a request.

In order to meet the above objectives, the *Integrated Services* (IS) model was proposed. This model supports both real-time and non-real-time applications. The underlying assumptions of IS model are [BCS94]:

- resources must be explicitly managed in order to meet application requirements;
- the service guarantees for a real-time service cannot be achieved without reservations;
- in spite of an application's ability to adapt, the end-to-end delay must be bounded;

- to secure statistical gain, it is desirable to use the Internet as an infrastructure for both real-time and non-real-time applications.

Quality of Service. Real-time applications based on their reaction to delay are categorized into two classes. First, applications that are intolerant to delay will have good performance only if they get a perfect upper-bound on the delay [Dav92]. Second, applications that do not require a perfect upper-bound on delay can adjust to variations in delay and are called adaptive applications. Based on these applications, two real-time service classes are defined: *guaranteed service* and *controlled load service*. The guaranteed service provides an upper bound on delay that is perfectly reliable, while the controlled load service provides one that is fair but not perfect.

The elastic applications always wait for the data to arrive. The main feature of these applications is that they typically use the arriving data immediately, rather than buffer it for later use. In case of delay, they always choose to wait for the data rather than proceed without it. An appropriate model for elastic applications is the existing best effort service class in the Internet.

Classes of Service. Based on the QoS requirement as quoted in the last paragraph, two real-time service classes, the guaranteed and the controlled load class, in addition to the best effort class can be introduced. The *Tspec*, which defines the data flow and *Rspec*, which defines the desired QoS, belong to the RSVP and are used to specify the service class completely. A separate subsection later describes RSVP.

The *guaranteed service* class provides firm end-to-end delay guarantees. This class provides both delay and bandwidth guarantees. A flow is described using a token bucket, a variation of the leaky bucket presented in the section on ATM. The Tspec here consists of a token bucket (r,b), a peak rate (p), a minimum policed unit (m) and a maximum datagram size (M). The token bucket has a depth b and a data rate r. The minimum policed unit (m) implies that any datagram of size less than m will be counted as being of size m. The Rspec is defined using a rate R and a slack term S in microseconds. The slack term quantifies the difference between the desired delay and that obtained by using a reservation rate R. Policing can be done in two ways. The first, *simple policing*, refers to comparing the arriving traffic against a Tspec. The second, *reshaping*, refers to the reconstruction of a traffic's shape to conform to the Tspec [SPG97].

The *controlled load service* is an enhanced quality of service to support applications requiring better performance than the best effort service. Service for this class is described by using only Tspec. As the network does not provide any quantitative guarantee, Rspec is not required. The parameters are the same as the guaranteed service class. For a time period T, the burst should not exceed $rT + b$. Otherwise, the packets violating this are considered as non-conforming [Wor97].

The *best effort* service class supports applications that do not have any deadlines. Examples of such applications include telnet and ftp. These applications are best suited for operation under the "best effort" service class and hence this class is maintained in the same state as it is in the current Internet model. This class has no Tspec or Rspec as there is no guarantee of any sort by the network.

Support for Service Classes. Once the service classes are in place, the question is, how can these classes be requested? By virtue of the multiple service classes, one can select a class suitable for the application. At the same time, all the network elements along the path of the flow such as routers must be informed of the flow and the parameters. Such a requirement can be realized by using RSVP. It is useful with a variety of QoS control services and deals with their setup mechanism. On the other hand, the Real Time Transport (RTP) protocol can be used to enforce end-to-end delay, which is a prime requirement for real-time applications. In addition, RTP can also monitor the QoS along with the delay. The next subsection deals with RTP separately.

Integrated Services over ATM. IP of the TCP/IP is interoperable today with different link layers such as ATM and Ethernet. Of the two, the ATM has multiple classes of service and also supports QoS functionalities. These characteristics help to support Internet Integrated Services over ATM. For this purpose, we need to look at mapping service classes and supporting RSVP over ATM.

The three service classes of IP service model can be mapped to five classes of ATM service model [GB97], as given in Table 3.

The most appropriate ATM service class for the corresponding IP service class is shown in italics. *RT-VBR* is best suited for guaranteed service as it can utilize the unused portion of other applications and can support more bursty sources. *NRT-VBR* is good for controlled load service since bandwidth and buffers can be allocated.

IP Service Class	ATM Service Class
Guaranteed Service	CBR or *RT-VBR*
Controlled Load Service	CBR, *NRT-VBR* or ABR
Best-Effort Service	*UBR*, Any ATM Class

Table 3: Mapping of Internet Integrated Service Classes to ATM Service Classes.

UBR is a natural choice for best effort service.

The RSVP messages used to set up flow reservations must follow the same IP path throughout the network. The main issue in supporting RSVP over ATM is the certainty of the timing of the setup of a virtual circuit for a specific QoS corresponding to RSVP. Other details are being discussed.

3.3.2 Real-Time Transport Protocol

In an inter-network, for playback of audio and video, TCP may be appropriate. However, for real-time delivery of audio and video, TCP and other reliable transport protocols are inappropriate due to the following reasons [Sch95]:

- TCP cannot support multicast.
- Even for unicast, reliable transmission is inappropriate for delay-sensitive data such as real-time audio and video. By the time the sender knows that the receiver has missed a packet and retransmitted it, a period of one round-trip time would have elapsed. In case of the TCP fast retransmit algorithm, three additional packetization intervals beyond the round-trip are needed.
- The TCP congestion control mechanisms decrease the congestion window when packet losses are detected (slow start). Audio and video have *natural* rates that cannot be suddenly decreased without starving the receiver.

Thus, there is a need for real-time protocols. In the following paragraphs we describe the *Real-Time Transport Protocol* (RTP) and its constituents.

RTP has been designed within the IETF [Sch95]. RTP provides end-to-end delivery services for data with real-time characteristics such as interactive audio and video.

The services provided include play-load type identification, sequence numbering, time stamping and delivery monitoring. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services; both protocols contribute parts of the transport protocol functionality. RTP supports data transfer to multiple destinations using multicast distribution if provided by the underlying network. RTP itself does not provide any mechanism to ensure timely delivery or provide other QoS guarantees, but relies on lower level services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. RTP does not address resource reservation.

The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence or determine the proper location of a packet. RTP consists of two parts: a *data part*, to carry data that has real-time properties and a *control part*, to monitor the QoS and to convey information about the participants in an ongoing session. RTP is a protocol framework, which can be integrated into applications rather than implemented as a separate layer.

Among the many use scenarios, the RTP level *Translators* and *Mixers* are notable [SCFJ96], [Sch95]. Suppose, in the case of a video conference, some participants have high-speed links while others have lower bandwidth links. Rather than sending low-bandwidth, low-quality video to all participants, or having each well connected participant generate both high and low bandwidth video, it may be preferable to install a video mixer at the low-bandwidth area. This *mixer* resynchronizes the incoming video packets to reconstruct the constant spacing generated by the sender, mixes these reconstructed video streams into a single stream, translates the video encoding into one of lower bandwidth and forwards the lower-bandwidth packet stream across the low-speed link.

Some of the intended participants in the video conference may be connected with high bandwidth links but might not be directly reachable via IP multicast. For example they might be behind an application-level firewall that will not let any IP packets pass. For these sites, mixing may not be necessary. In such cases another type of RTP-relay called a *translator* may be used. Two translators are installed, one on either side of the firewall, with the outside one funnelling all multicast packets received through a secure connection to the translator inside the firewall. The translator inside the firewall sends them again as multicast packets to a multicast group restricted to the site's internal network.

RTP Data Packets. RTP data packets consist of a 12 byte header followed by the payload. The main fields are given below.

Payload type. A one byte payload type identifies the kind of payload contained in the packet; for example, JPEG video. Having payload type in every packet avoids connection setup and permits dynamic changes of encodings.

Timestamp: A 32-bit timestamp describes the generation instant of the data contained in the packet. The timestamp frequency depends on the payload type. Several video packets may have the same timestamp if they belong to the same video frame.

Sequence Number. A 160 bit packet sequence number allows loss detection and sequencing within a series of packets with the same timestamp.

Marker bit. The interpretation of the marker bit depends on the payload type. For video it marks the end of a frame, for audio the beginning of a talkspurt.

Synchronization Source Identifier (SSRC). The SSRC identifies the synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC identifier.

Contributing Source Identifier (CSRC). The CSRC lists the contributing sources for the payload contained in this packet. CSRC fields are inserted by the mixers using the SSRC identifiers of contributing sources.

The RTP Control Protocol (RTCP). RTCP is based on the periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the data packets. RTCP performs four functions:

1. The primary function is to provide feedback on the *quality* of data distribution. The information is also used in the flow and congestion control functions.

2. RTCP carries a persistent transport level identifier for an RTP source called the *canonical name* or CNAME. The receivers require the CNAME to keep track of each participant in case a conflict is discovered or a program is restarted. Another function is to synchronize the clocks of different media streams from the same source as in the case of audio and video synchronization.

3. RTCP packets are sent periodically by each member and it is necessary that the rate be controlled in order for RTP to scale up to a large number of participants. Since every participant sends control packets to all others, each can independently observe the number of participants.

4. An optional function is to convey minimal session control information; for example participant identification to be displayed in the user interface. This is more likely to be useful in cases where the members join and leave without membership control. Though RTP serves as a convenient channel to reach all participants, it is not expected to support all communication requirements of an application.

In RFC 1889 [SCFJ96] several RTCP packet types have been defined to carry a variety of control information:

- *SR.* Sender report, for transmission and reception statistics from participants that are active senders.
- *RR.* Receiver report, for reception statistics from participants that are not active senders.
- *SDES.* Source description items, including CNAME.
- *BYE.* Indicates end of participation.
- *APP.* Application specific functions.

3.3.3 Resource Reservation Protocol

The Internet in its present form offers a very simple service model, which can be best described as point-to-point best effort service. The best effort service has worked well for the traditional traffic consisting of electronic mail, file transfers, etc. The emerging real-time traffic is comprised of applications which include remote video and multimedia conferencing among others. The Internet's primitive service model fails to address two main *requirements* of these new applications [ZDE+93]. First, many applications are very responsive to the quality of service offered by the underlying networks to their data. A network, to offer adequate QoS, has to go farther than the best effort service model and permit traffic streams to reserve resources. Second, the emerging new applications are not solely point-to-point. They are often multipoint-to-multipoint. Such applications include interactive multimedia conferencing, where each one of the participants sends and receives data.

Given the Internet's lack of support for the newer applications, there is a spur in research activity aimed at developing new architectures. Resource reservation is

one of the components in such architectures. In the following paragraphs, we briefly present the functionality of *Resource ReSer Vation Protocol* (RSVP), which is designed for an integrated services Internet [BZB+96].

RSVP is used by a host, on behalf of an application data stream, to request specific QoS from the network of a particular data stream or flow. RSVP is also used by the routers to deliver QoS control requests to all nodes along the path(s) of the flows and to establish and maintain state to provide the requested service. RSVP requests will generally, although not necessarily, result in resources being reserved in each node along the data path. The *attributes* of RSVP can be summarized as follows [BZB+96]:

- RSVP makes resource reservations for both unicast and many-to-many multicast applications, adapting dynamically to changing group membership as well as to changing routes.
- RSVP is simplex, i.e., it makes reservations for unidirectional data flows.
- RSVP is receiver-oriented, i.e., the receiver of the data flow initiates and maintains the resource reservation used for that flow.
- RSVP maintains "soft state" in the routers, providing graceful support for dynamic membership changes and automatic adaptation to routing changes. Soft state is the control state in the hosts and routers that will expire if not refreshed within a specified amount of time.
- RSVP is not a routing protocol but an Internet control protocol and it depends on present and future routing protocols.
- RSVP transports and maintains opaque state for traffic control and policy control.
- RSVP provides several reservation models or styles, defined below, suitable for a variety of applications.
- RSVP provides transparent operations through routers that do not support it.
- RSVP supports both IPv4 and IPv6.

Sender Selection	Reservation	
	Distinct	Shared
Explicit	Fixed-Filter Style	Shared-Explicit (SE) Style
Wildcard	(None Defined)	Wildcard-Filter (WF) Style

Figure 5: Reservation Attributes and Styles

Reservation Model. An elementary RSVP request consists of a *flowspec* and a *filter spec.* Together, they are a *flow descriptor.* The flowspec specifies a desired QoS. The filter spec, together with a session specification, defines the set of data packets – *the flow* – to receive the QoS defined by the flowspec. The flowspec is used to set parameters in the node's *packet scheduler*, while the filter spec is used to set parameters in the *packet classifier.* Both packet classifier and packet scheduler are traffic control functions in the primary data packet forwarding path. The packet classifier selects a service class for each packet in accordance with the reservation state setup by RSVP. While the packet scheduler implements QoS for each flow, it uses one of the service models defined by the Integrated Service Working Group [BZB+96]. The flowspec in a reservation request will generally include a service class and two sets of numeric parameters; an Rspec (R for reserve) that defines the desired QoS and Tspec (T for traffic) that describes the data flow [BZB+96].

Reservation Styles. A reservation request includes a set of options that are collectively called the reservation style, shown in Figure 5.

One reservation option concerns the *treatment* of reservation for different senders within the same session: establish a distinct reservation for each upstream sender, or make a single reservation that is shared among all packets of selected senders.

Another reservation option *controls* the selection of senders; it may be an explicit list of all selected senders or a wildcard that implicitly selects all the senders to the session. In an explicit sender selection reservation, each filter spec must match exactly one sender, while in a wildcard sender selection no filter spec is needed.

RSVP Message Types. There are two fundamental RSVP message types : *Resv* and *Path* [BZB+96].

Resv. Each Receiver host sends reservation request (Resv) message upstream towards the sender(s). These messages create and maintain *reservation state* in each node along the path(s). Resv messages must finally be delivered to the sender hosts themselves, so that the hosts can set up appropriate traffic control parameters for the first hop. At each intermediate node two general actions are taken on request:

1. *Make a reservation.* The request is passed to admission control and policy control. If either test fails, the reservation is rejected and RSVP returns an error message to the appropriate receiver(s). If both succeed, the node uses the flowspec to set up the packet scheduler for the desired QoS and the filter spec to set the packet classifier to determine the QoS class for each of the data packets.

2. *Forward the request upstream.* The reservation request is propagated upstream toward the appropriate senders. The set of sender hosts to which a given reservation request is propagated is called the "scope" of that request.

Path. Each RSVP sender host transmits Path messages downstream along the unicast or multicast routes provided by the routing protocol(s) along the paths of the data. These path messages store path state, which contains at least the unicast IP address of the previous hop node, in the node along the way. The additional information in Path messages and their uses are:

1. *Sender template.* A Path message is required to carry a sender template, which describes the format of packets that the sender will originate. This template is in the form of the filter spec that would be used to select this sender's packet from others in the same session on the same link.

2. *Sender Tspec.* A Path message is required to carry a Sender Tspec, which defines the characteristics of the data flow that the sender will generate. This spec is used by traffic control to prevent over-reservation, and perhaps unnecessary admission control failures.

3. *Adspec.* A Path message may carry a package of One Pass With Advertisement (OPWA) advertising information, known as an Adspec. An Adspec received in a Path message is passed to the local traffic control, which returns an updated Adspec. The updated version is then forwarded in path messages sent downstream. The result is that the Adspec at the receiver end may be used to gather information that may be used to predict the end-to-end QoS.

3.4 Packet Scheduling

Computer networks facilitate user sharing or multiplexing of resources such as printers, data archives, links and sites on the World Wide Web. The nature of sharing naturally introduces the problem of contention for the shared resource. The server managing the request employs a scheduling scheme to resolve the contention issue. A scheduling scheme basically decides which client is to be served next. A scheduling scheme consists of two components. First, it determines the order of service for the clients. Second, it manages the service queue in which requests are waiting. Scheduling disciplines are required as the networks are expected to carry diverse traffic consisting of best effort and guaranteed service applications. Apart from the best effort traffic, future networks are slated to carry applications requiring bounds on their performance.

A scheduling discipline needs to satisfy four *requirements* [Kes97]. *First*, it should be simple to implement for both guaranteed service and best effort service. *Second*, it needs to maintain fairness and protection for best effort connections. *Third*, it needs to manage performance bounds for guaranteed service connections. *Finally*, it should possess a simple and efficient admission control. In the next two subsections, we briefly present some scheduling mechanisms for the best effort and guaranteed service connections.

3.4.1 Scheduling Best Effort Connections

Best effort applications expect the scheduling principle to provide max-min fair allocation of resources. This can be formally stated as follows [Kes97]:

- Allocation of resources proceeds in the order of increasing demand.
- No source gets a resource share larger than its demand.
- Sources with unsatisfied demands get an equal share of the resource.

In the following paragraphs we present some scheduling disciplines. Additional schemes and further information can be found in [Kes97].

Generalized Processor Sharing Generalized Processor Sharing (GPS) assumes that packets are in separate logical queues [Kes97]. During a finite interval it visits

each non-empty queue and serves an *infinitesimally* small amount of data from each queue. The idea is to visit every logical queue at least once in the finite interval. Weights can be associated with connections, which will allow each connection to receive service proportional to its weight whenever its queue is not empty. In case of an empty queue the scheduler skips to the next non-empty queue. It is claimed that this will result in a max-min fair share of the bandwidth allocation.

Weighted Round Robin Weighted Round Robin (WRR) is the simplest emulation of the GPS scheme. The main difference is that instead of serving an infinitesimally small amount of data, WRR serves a packet from each non-empty queue. WRR emulates GPS reasonably well when both the weights and packet sizes are same. Connections with different weights are served in proportion to their weights. This scheme is used for best effort traffic.

Weighted Fair Queueing Weighted Fair Queueing (WFQ) along with packet-by-packet generalized processor sharing approximates the GPS scheme. These two schemes do not make the "infinitesimally small amount of data" assumption and with variable packet sizes do not need to know the mean packet size of the connection in advance. The idea behind WFQ is to determine the time a packet would complete service if it were to be served with a GPS server and then serve packets in the order of these finishing times.

3.4.2 Scheduling Guaranteed Service Connections

WFQ presented in the previous subsection can be used both for best effort and guaranteed service connections. In order to provide performance guarantees, a bound on the worst case end-to-end delay is determined. The next two paragraphs contain two notable mechanisms [Kes97].

Virtual Clocks. A virtual clock scheduler is an independent scheme. The packets are stamped with a tag, which determines the serving order. The tag allocates the virtual transmission time [SN95]. One distinguishing feature is that the tag values do not emulate the GPS discipline as in the case of WFQ. The virtual clock scheduler emulates time division multiplexing in a similar way that a fair queueing scheduler

emulates GPS. Both the virtual clock scheduler and WFQ provide similar performance when all connections are backlogged. But, when two connections are backlogged, the virtual clock scheduler does not provide a fair performance.

Rate Controlled Scheduler. Rate Controlled Scheduler (RCS) schemes can provide bounds on the connection bandwidth and delay. RCS consists of two components; a *regulator* and a *scheduler*. The regulator houses the incoming packets, which determines the eligibility of a packet with the help of one of many algorithms. Upon eligibility, the packet is sent to the scheduler which arbitrates among eligible packets. The flow of packets can be controlled by the regulator operating under some constraints, such as the arrival of the packets to the scheduler at a rate smaller than that of the peak rate. On the other hand, the scheduler can emulate many disciplines by issuing directions to the regulator such as the restriction of the number of eligible packets that arrive from the regulator. Thus, with a proper choice of regulator and scheduler, a variety of performance bounds can be achieved.

3.5 ATM and Internet

In the current Internet framework, the majority of constituent networks run TCP/IP. In order to use the existing Internet frame-work in the future, it is essential that both the ATM and Internet technologies interoperate. We detail an effort in this direction led by the IETF's IP over the ATM working Group, in the following subsections.

3.5.1 TCP/IP Over ATM

Transmission Control Protocol/Internet Protocol (TCP/IP) is widely used in the networking world [CCGP95]. TCP/IP places emphasis on inter-networking using a *connectionless* model. In TCP/IP, the IP layer is responsible for moving packets from one node to another, while TCP is responsible for verifying the correct delivery of data from client to server. TCP/IP performs well on traditional services like file transfer, remote login and e-mail. TCP/IP, in its basic form, is *not suitable* for real-time applications for the following reasons:

1. IP packets are based on the datagram model and are not suitable for real-time traffic.

2. Routing of IP packets via a connectionless model may lead to out of order delivery. Also, media synchronization is an accompanying problem.

3. Since the delivery is best effort only, there is no support for QoS.

Currently, the Internet is growing at an extremely fast pace. Within the Internet, a large number of networks run TCP/IP. Thus, a natural question is whether or not the TCP/IP can run over ATM? The answer is positive and, as a result, the IETF initiated the *IP over ATM Working Group* to develop standards for routing and forwarding IP packets over ATM subnetworks. IP over ATM project issues are listed below [CCGP95]:

1. The encapsulation and transmission of IP network or link layer packets across an ATM Adaptation Layer 5 connection.

2. The provision of a mechanism for the resolution of IP addresses to their corresponding ATM addresses. This is known as the *Classical IP over ATM* protocol and solves the problem of IP unicast traffic over ATM. The multicast part uses Resource ReSerVation Protocol and the Internet Integrated Services model.

3. The expansion of the Classical IP over ATM model to include the emerging *Next Hop Resolution Protocol*, described later in this section.

4. The integration of multimedia services into IP, which will also allow connectivity to ATM networks.

3.5.2 IP over ATM Encapsulation

Transporting IP packets within the ATM cells needs a standard governing encapsulation. Such encapsulation allows for the multiplexing of multiple packet types at the network layer on the same connection. This results in connection resource conservation and reduces the connection setup time. The multiplexing field allows a node that receives a network layer packet across an ATM connection to know what type of packet has been received and to know about the application by which it must be received. The methods identifying this type of information are [CCGP95]:

1. *Virtual Channel* multiplexing carries only one protocol across the ATM connection with the type of protocol identified at the connection setup time.

2. *Logical Link Control/SubNet Access Protocol* (LLC/SNAP) can carry many protocols over the same VC connection. In this case, the type of packet is identified by a header.

3.5.3 IP over ATM Address Resolution

The "classical" view of IP is one in which clusters of IP nodes consisting of hosts and routers with similar subnet addresses are connected to nodes outside their cluster by IP routers [CCGP95]. The IETF adhered to this "classical" view by grouping IP nodes into *Logical IP Subnets* (LIS). The nodes in an LIS share the same IP subnet and communicate outside through an IP router. A node, upon joining an LIS, establishes a connection with the ATM-Address Resolution Protocol (ATMARP) server, which has information about the server address. The ATMARP server detects the connection and sends Inverse ARP request to the attached client, asking for the IP address with the knowledge of the ATM address. The server stores the ATM and IP addresses in the ATMARP table by requesting them. There is also a proposal to have the server learn this information by observing client messages.

The destination address is resolved by a client sending an ATMARP request to the server, requesting the ATM address by providing the IP address of the destination host. The server responds with an ATMARP reply if the address mapping is found. Otherwise, it returns a negative acknowledgement. The ATMARP server sends periodic Inverse ARP requests to update its addresses. In case there is no response from a client, it is deleted from the ATMARP address table.

The *Next Hop Resolution Protocol* (NHRP) is used to establish direct connection between two clients when they belong to the same ATM network [LKPC97]. NHRP consists of an NHRP server (NHS), NHRP client (NHC) and the protocol between them. Each LIS contains at least one NHS. Every end system in a LIS is a NHRP client. NHRP aids transmission mechanisms like ATM, X.25 and Frame relay. The clients can be divided into administrative domains. NHRP applies within a single domain and can provide access to entry points of other domains. In this protocol NHS is employed. The NHS stores the IP to ATM mappings. The prefixes of IP address are accessed from the routers serving the NHS. The nodes register with the NHS using a registration packet. The NHS has a list of destinations served by other NHS's. For routing, the client sends its request to its NHS asking to resolve an ATM

address. If the address is being served by the NHS it returns the mapping. Otherwise it forwards the request to another NHS. The reply is sent along the reverse route. Any further requests are responded to without forwarding the request. The NHRP is likely to be used on routers for frame relay and X.25.

The future work on IP over ATM domain includes support for multicast of IP packets over ATM and multiprotocol encapsulation over AAL5.

3.6 Discussion

Real-time applications exhibit stringent timing conditions, which are crucial to achieve intended performance levels. The medium of our focus, video, is accompanied by the dynamic nature of frame sizes along with critical timing requirements. Owing to the large frame sizes, video is handled almost completely in a compressed form. It is the varying levels of compression that give rise to the dynamic nature in frame sizes. This is evident in case of I, P and B frames of MPEG sequences where, at times, the size of an I frame can be easily 10 times that of a B frame. The timing constraints arise because of the need to maintain continuity in playing frames. Otherwise, the artifacts become noticeable, and thus would lead to degraded performance. For example, an MPEG sequence operating at 30 fps needs a frame to be played every 33 ms. Though the eye can tolerate a delay of 40 ms, any delay beyond that would impair the quality [SN95]. In order to meet such requirements, we need mechanisms to guarantee required performance levels before embarking with the actual transmission. ATM is one such technology, which offers the QoS mechanism.

Much of the future success of ATM is attributed to ATM's ability to interoperate with IP. But, there are two major *differences* between ATM and Internet. First, ATM is connection-oriented, wherein the two parties establish a connection before transmission. On the other hand, IP is connection-less wherein each packet is forwarded by routers independently on a hop-by-hop basis. This can be solved in two ways. One possibility is to establish a new connection on demand. However, this is not viable in case of small amounts of data transfer due to the high cost of setting up and tearing down the connection. The other option is to forward the data through preconfigured connection(s). This scheme may not be the optimal path and the amount of data could overwhelm the connection. The second major difference is that of ATM's QoS

facility and IP's best effort service. This is being solved by extending the Internet as explained in the Integrated Internet section.

Networks offering QoS need packet scheduling schemes, which can help sustain the promised QoS. The work by [SL97] proposes such a scheme wherein the authors enhance the GPS scheme by addressing its shortcomings. In GPS, service rate received by a session is proportional to the backlogged sessions in the system. If another session becomes active, the service rate of a session may change abruptly. This could result in increased delay for consecutive packets. The authors propose a new scheme which alleviates this problem called *Slow-Start GPS*. In this method, when a new session becomes active, instead of giving its share of bandwidth immediately, it is allocated gradually from 0 to final value. In this way the "older" sessions do not experience a sudden drop in service rates.

In another work, the authors [APKR95] experiment with the traffic control mechanism of ATM; namely the Usage Parameter Control (UPC) aided by leaky buckets. In this mechanism, the authors demonstrate the control functions for a connection defined by PCR and SCR. They further illustrate controlling the PCR of a video connection by using the PCR's of the connection and the source in addition to Cell Delay Variation (CDV) tolerance. It is observed that whenever the actual PCR exceeds contracted PCR, any change in CDV tolerance has little effect on the Cell Discard Ratio. Once the CDV tolerance is larger than the the inverse of contracted PCR, the UPC discards exactly that excess amount of traffic. In case the actual PCR is below the contracted PCR, only cell clumping will cause cell discards. Given such a scenario wherein losses occur, new mechanisms are required so that the quality of video can be maximized at receiver end.

Some mechanisms can be proposed, which can help maximize the visual quality. For example, priority can be established for video frame drops under constrained bandwidth conditions. As mentioned earlier, MPEG contains 3 types of frames of varying importance. Such information can be used to drop the frames in decreasing order of importance; first B, then P and finally I. This could lead to a considerable gain instead of loosing all I frames which would seriously damage the set of P and B frames dependent on them. In order to recover from packet losses, error correction techniques like FEC are used. FEC is preferred since the traditional techniques would contribute delay which the real-time applications cannot tolerate. FEC is supported by ATM in AAL1. However, FEC comes with a trade off [Yas96]. It facilitates recovery of lost

information at the cost of additional bandwidth required for transmission of correction information. Due to the issues involved, FEC is till a topic of discussion.

Chapter 4

Problem Statement

We consider the transmission of multiple video streams, such that a bandwidth reservation is shared between them. We are interested in a case when the bandwidth available is not sufficient to ensure zero-loss transmission, i.e., the bandwidth is less than the sum of average bandwidths of the participating video streams. In this environment, we wish to investigate the effect of bandwidth and buffer management policies on the quality of video streams received at the destination.

Accordingly, in this chapter, we detail our approach towards addressing this problem. We first present the model employed in our study. We then provide information about the video sequences used in our simulations. Then, we cover the simulation environment and next, we discuss the statistics-gathering process and highlight the performance measures. In the last section, we present the effect of bandwidth variation on the performance of each sequence.

4.1 The Model

Our model consists of a video gateway serving a link of fixed bandwidth. The video gateway has a finite number of buffers, small enough to ignore the delay due to queueing. We have multiple sources generating video data in the form of frames. Each frame of video-data is characterized by its size. The data arriving at the gateway is subjected to the *Frame Admission Control* (FAC) before being buffered for transmission as in Figure 6. The FAC implements the buffer management policy at the frame level. FAC operates with frames instead of the conventional packets

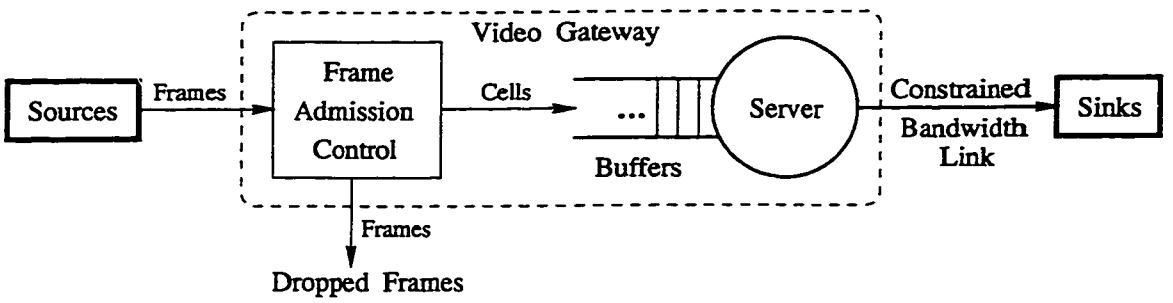


Figure 6: Network Model

as we deal with frame-drops. The buffer size is quantified in terms of ATM cells. Frames in the queue of buffers are forwarded in the form of ATM cells on the outgoing link. The constrained bandwidth of the outgoing link reflects the negotiated reserved bandwidth with the network.

The FAC is the key component in our model, and determines the quality of video transmitted. Any frame that is not dropped at the FAC is considered to be received at the sink.

4.2 Video Sequences Characterization

We call a set of video frames a video sequence or a video stream. For the experiments, we use MPEG-1 sequences consisting of only I-frames obtained from the Internet. Initially, we determine the frame sizes in bytes by parsing video sequences. We repeat video sequences in order to obtain sufficient frames for the entire simulation length. We then convert each frame size in terms of ATM cell's payload of 48 bytes. The resulting frame sizes in ATM cells are used as input data for the experiments. We characterize video sequences with the following: *minimum frame size*, *maximum frame size*, *average frame size* and *standard deviation of frame sizes*. Table 4 lists the characteristics of video sequences employed in the simulations.

We present the frame size characteristics of ten sequences employed in our simulations in Figures 7 through 16. These video streams reflect different information content. Sequences `bla`, `jet` and `can` have fast motion pictures. Among these three, `bla` has a large number of scene changes. Streams `foo` and `sum` are live recordings of a football match and sumo wrestling respectively. These two streams also show fast motion between consecutive pictures. In contrast, `tha` and `suk` cover low activity

Streams	Maximum	Minimum	Average	Std. Deviation
bla	35.9	15.9	26.9	151.4
boo	68.4	27.2	43.2	568.7
can	69.5	28.4	48.4	269.2
jet	74.1	15.1	46.7	824.5
mov	61.5	55.1	58.8	77.8
tha	34.9	6.5	23.4	293.4
suk	60.6	33.8	42.0	296.1
sum	126.5	30.4	85.1	1276.5
foo	116.2	7.6	92.7	1517.3
boe	75.3	20.6	41.7	507.8

Table 4: Frame Size Characteristics (in ATM cells) for Ten Video Streams

fast motion between consecutive pictures. In contrast, tha and suk cover low activity images of a total solar eclipse and a parked aircraft respectively. The last three video clips named boo, boe and mov are captured with a moving camera showing different levels of activity between the pictures. With a collection of these sequences, we intend to capture a wide range of content and frame size variation. As the graphs of frame size characteristics illustrate, every sequence shows a different spread in terms of their frame size characteristics.

4.3 Simulation Environment

In this section we first present details of the simulator used. We then list the assumptions under which our experiments are conducted, followed by the set of configuration parameters.

4.3.1 The *ns* simulator

We enhance the *ns* version 1 simulation tool [MF95] developed by the Network Research Group at the Lawrence Berkeley National Laboratory for our experiments. *ns* is easily configurable and has a programmed event-driven simulation engine. A network topology is realized in *ns* by using three primitive building blocks: nodes, links and agents. Nodes act as containers for agents, the active objects that drive the

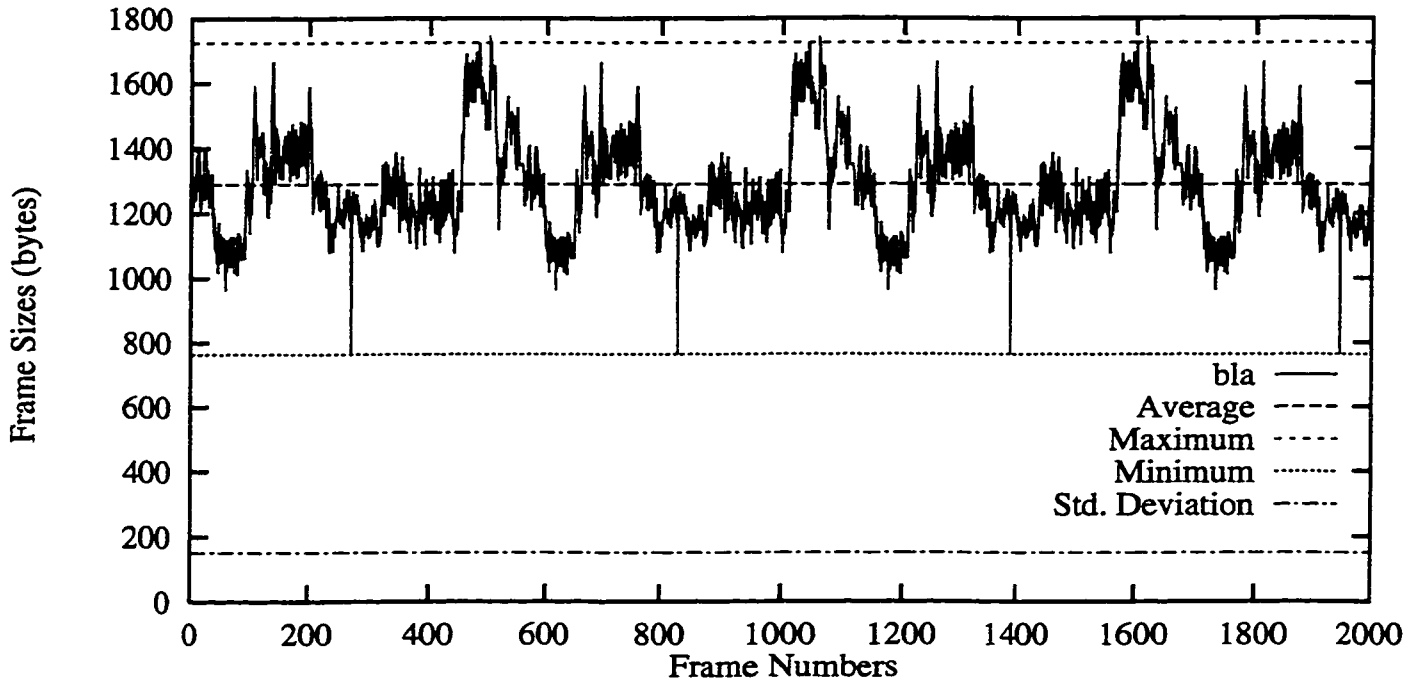


Figure 7: Frame Size Distribution of bla Sequence

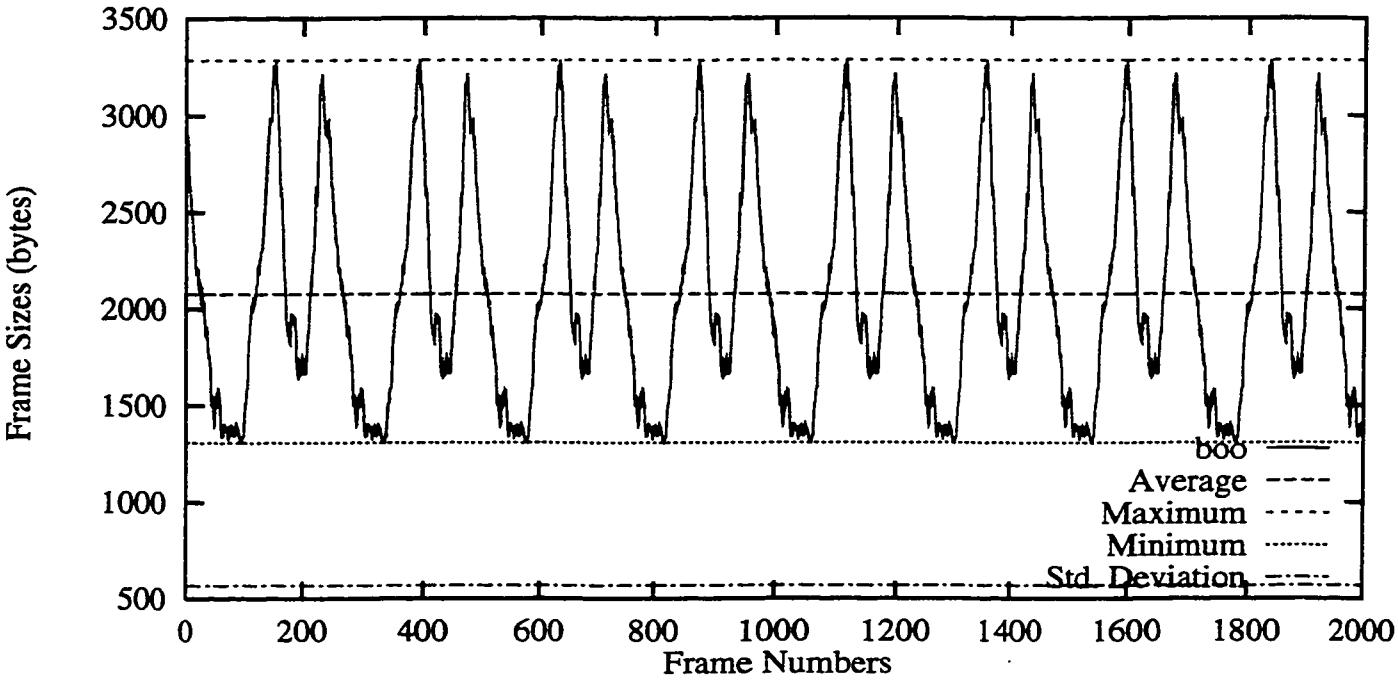


Figure 8: Frame Size Distribution of boo Sequence

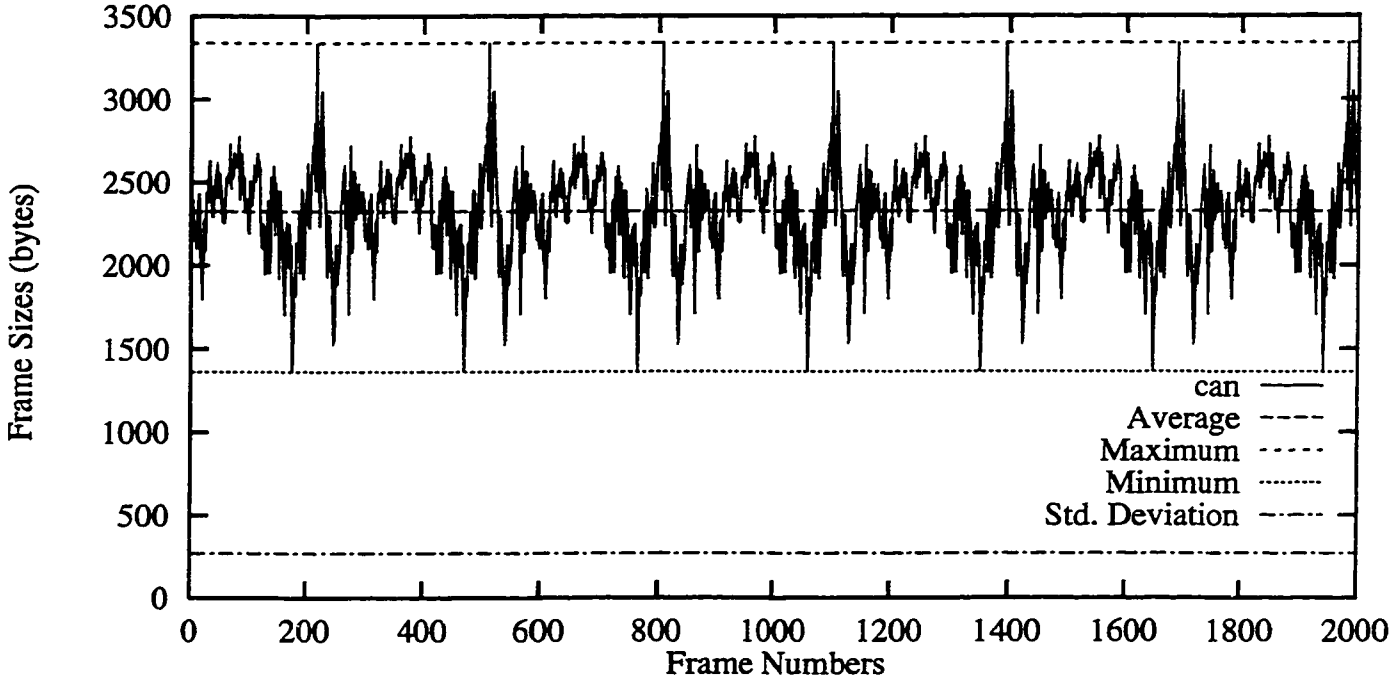


Figure 9: Frame Size Distribution of can Sequence

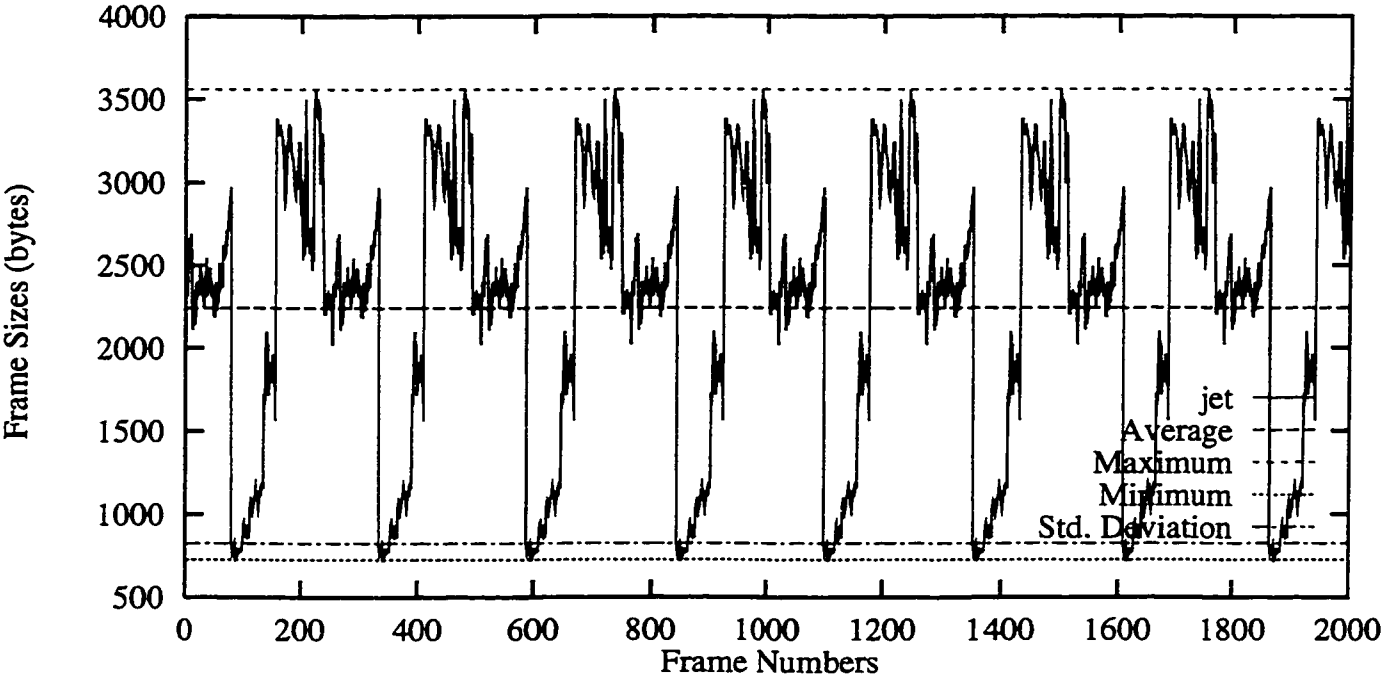


Figure 10: Frame Size Distribution of jet Sequence

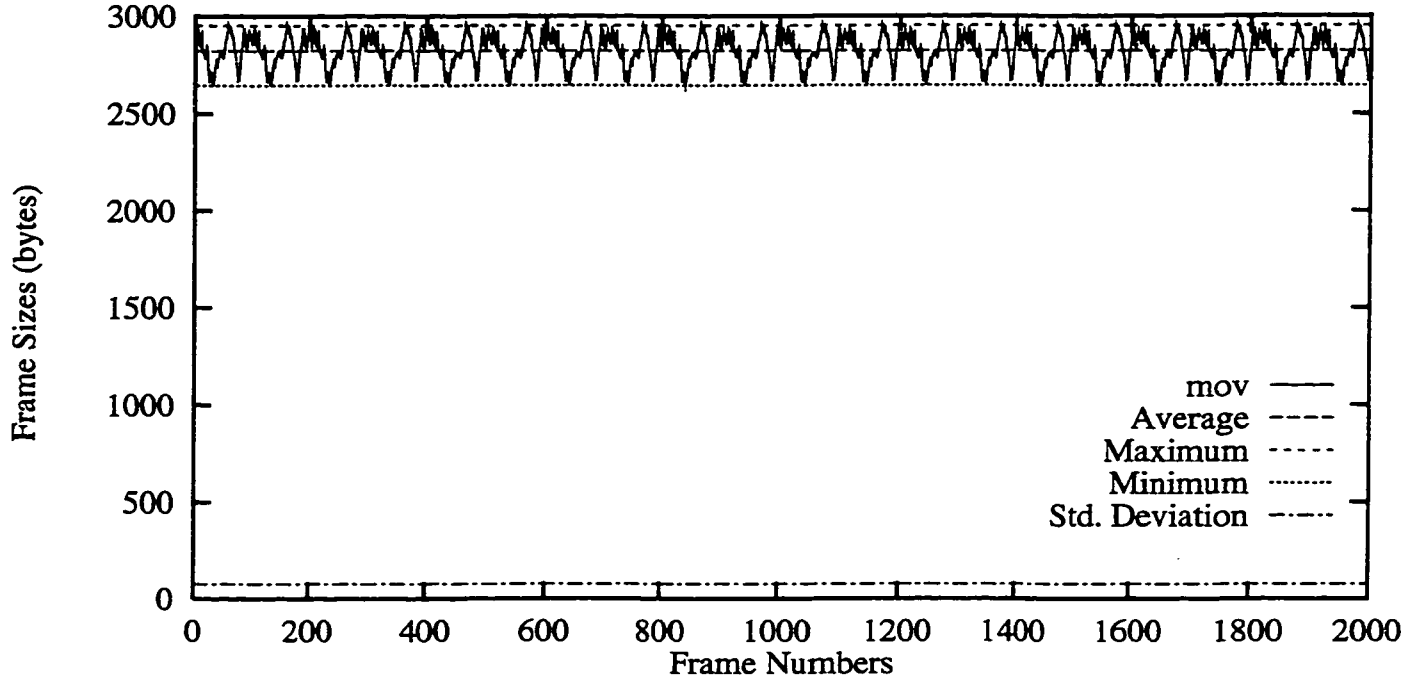


Figure 11: Frame Size Distribution of `mov` Sequence

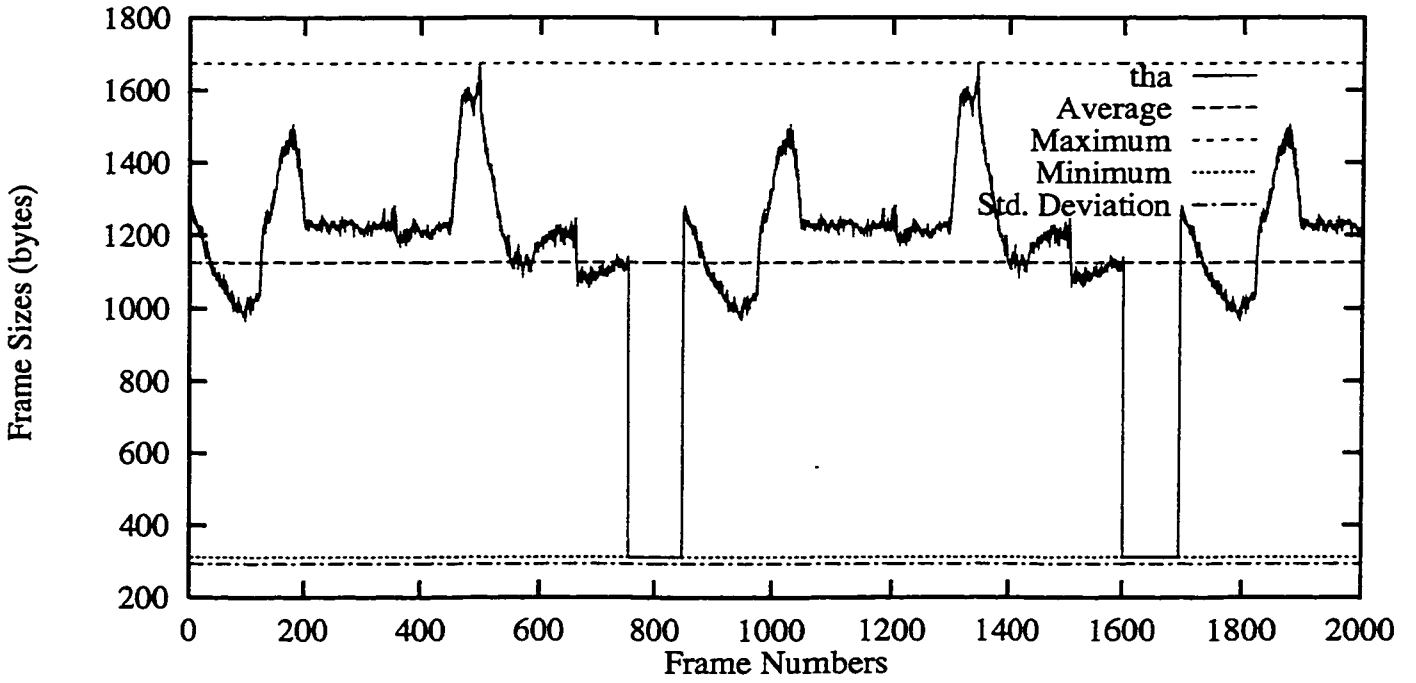


Figure 12: Frame Size Distribution of `tha` Sequence

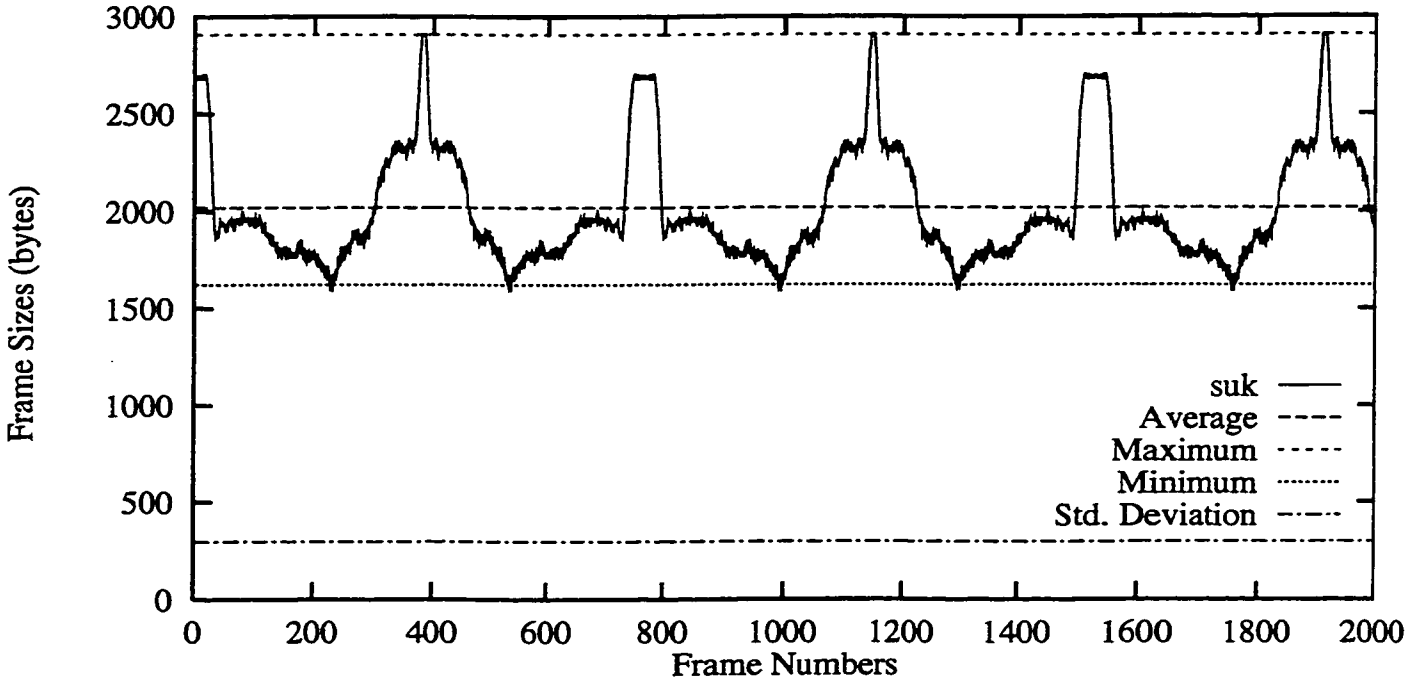


Figure 13: Frame Size Distribution of suk Sequence

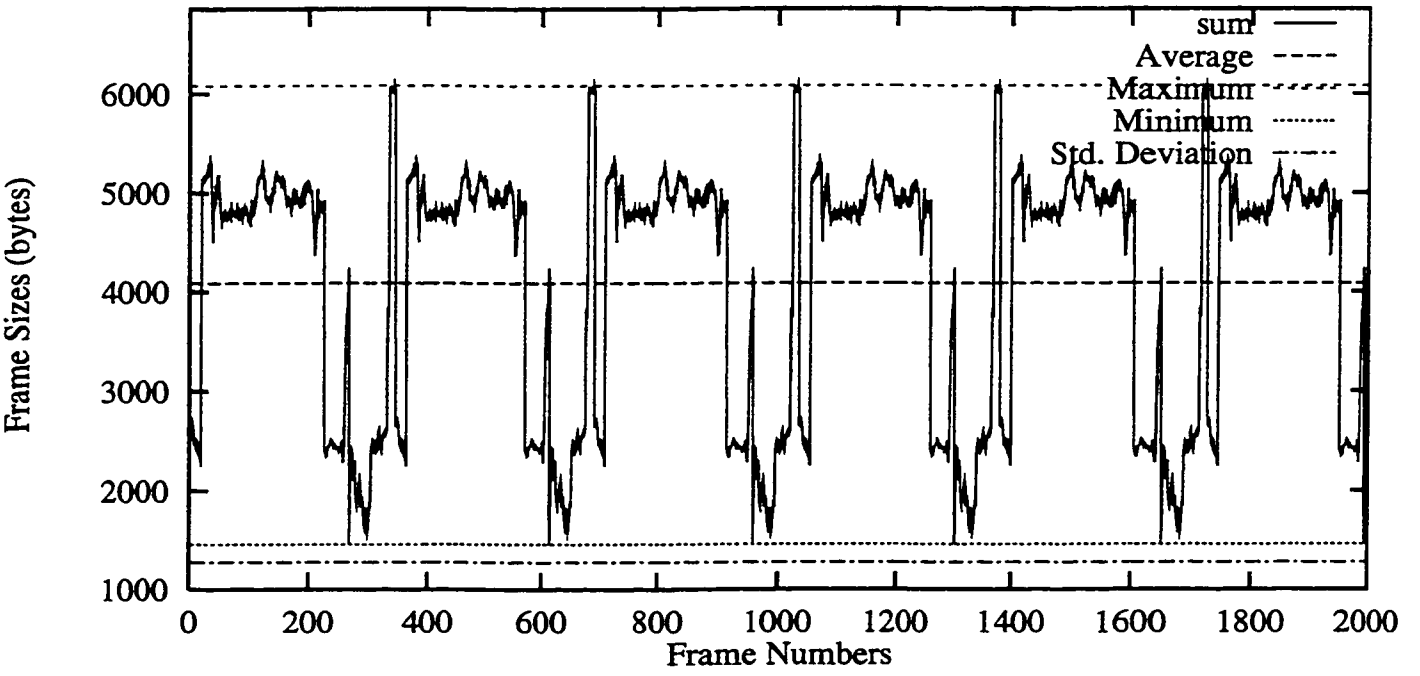


Figure 14: Frame Size Distribution of sum Sequence

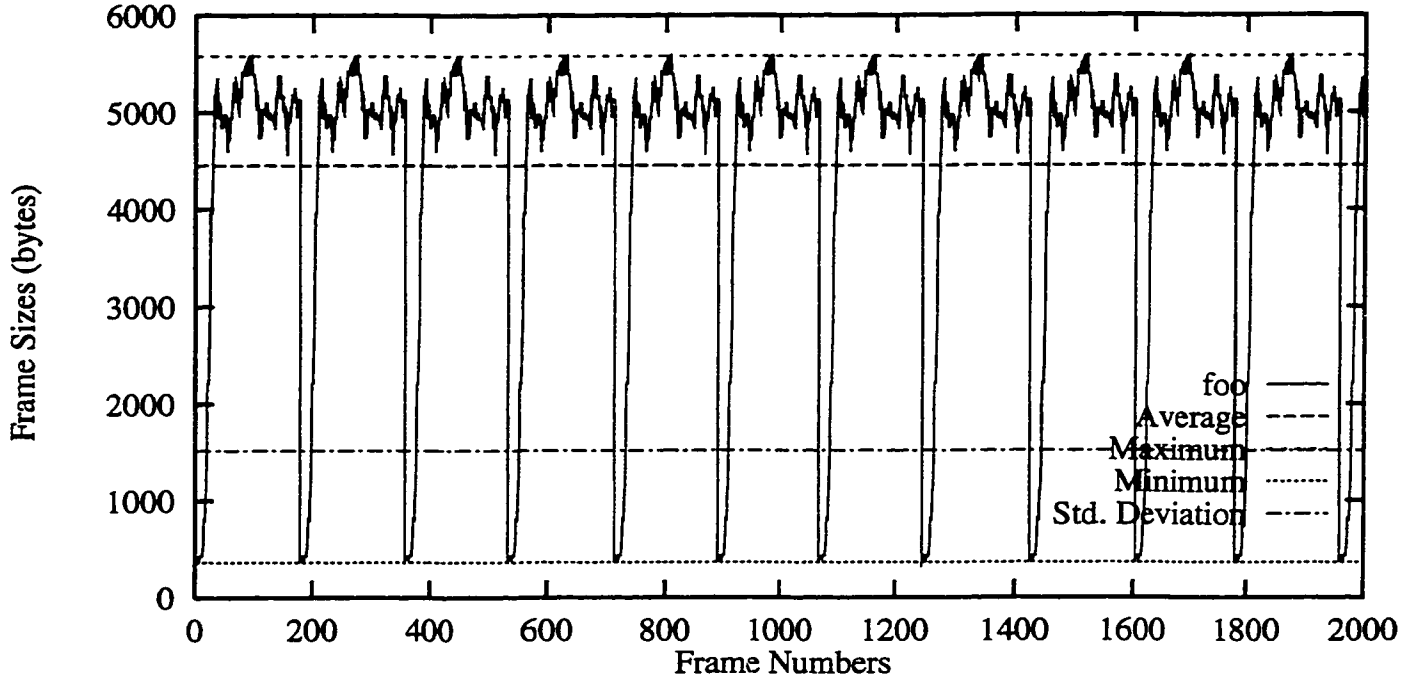


Figure 15: Frame Size Distribution of foo Sequence

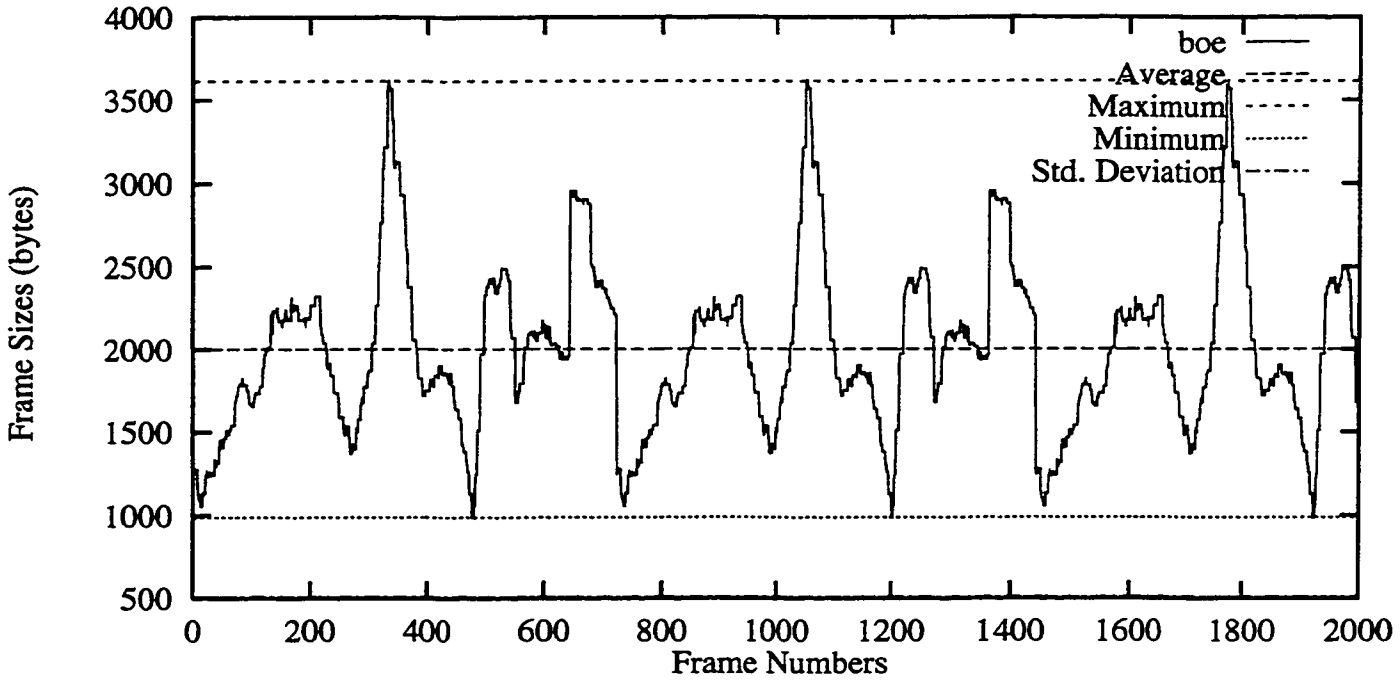


Figure 16: Frame Size Distribution of boe Sequence

simulation. Sources and sinks are defined with the help of nodes and are connected using the link command. The *ns* simulation description language is an extension of the Tool Command Language, (Tcl) [Ous94]. A simulation is defined by a Tcl program. We have upgraded this simulator to support video traffic. For this purpose, we have introduced video specific sources and sinks, along with necessary changes to implement FAC.

4.3.2 Simulation Set-up

Sources generate frames, based on the natural frame rate, which is fixed in our simulations at 30 fps. Frames arriving at the gateway are first checked by the FAC. It decides whether the frame is admissible or not. If the frame is admissible, the frame is sent to the queue, otherwise it is dropped. At the sink, we record the arrival status of frames and further process them through perl scripts [WCS96] to generate the output information.

The link capacity reflects a value in the constrained bandwidth region. For this purpose, we employ reduced values of the sum of average bandwidths of all the ten sequences, called *Average Bandwidth.* We determine the *Average Bandwidth* based on the *Average Rate* of ten sequences. The *Average Rate* for a particular sequence is computed as:

$$Average\ Rate\ (bps) = Frame\ Rate\ (fps) \times\ Average\ Frame\ Size\ (bits)$$

The *Average Bandwidth* is obtained as:

$$Average\ Bandwidth\ (bps) = \sum_{seq=1}^{10} Average\ Rate_{seq}\ (bps)$$

4.3.3 Assumptions

The assumptions employed in our simulations are as follows:

- The video frame sizes are known apriori. This assumption is used in computing the bandwidth values as explained in the previous section.
- There is no delay due to the frame admission part of FAC.

- Frame rate is fixed at 30 fps. The arrival times of the frames may vary if the frame arrivals are desynchronized.
- All packets (ATM cells) belonging to a particular frame will be available at their respective arrival time at the queue.

4.3.4 Frame Arrival Process

The basic form of frame arrivals in our simulation environment, is guided by synchronized arrivals. In this process, all the sources start at the same time and generate frames with an interval equivalent to a frame period (also called inter-arrival period). We define a *frame period* as the inverse of a *Frame Rate.* As a result of this process, all frames corresponding to the ten sources arrive at the same time at the gateway, thus leading to a highly synchronized arrival pattern. In other words, such arrivals contribute to frame clumping at the queue. This leads to a serious contention for buffer space. In this situation, frames belonging to the first few sequences are almost always accepted into the queue, while the sequences at the end experience losses. Thus, it is necessary to alter the arrival pattern to break the high degree of synchrony.

We desynchronize every frame arrival in order to break the high degree of synchronization. We call such an arrival pattern *desynchronized frame arrival.* This type of arrival process reflects the real-world network environments wherein the artifacts lead to variations in transmission time. We affect desynchronization of frame arrivals by adding some randomness to arrival times. Thus, in our simulations the range over which the frame arrivals can vary is chosen to be equivalent to the frame period. As a result of this interval, two consecutive frames can arrive either with a gap of twice the inter-arrival time (66 ms) or back-to-back. The inter-arrival time is used in specifying the randomness interval for obvious reasons, as it is the smallest identifiable unit in video transmission.

4.4 Results Generation

We first describe the statistics gathering process and conclude with a description of performance measures in this section.

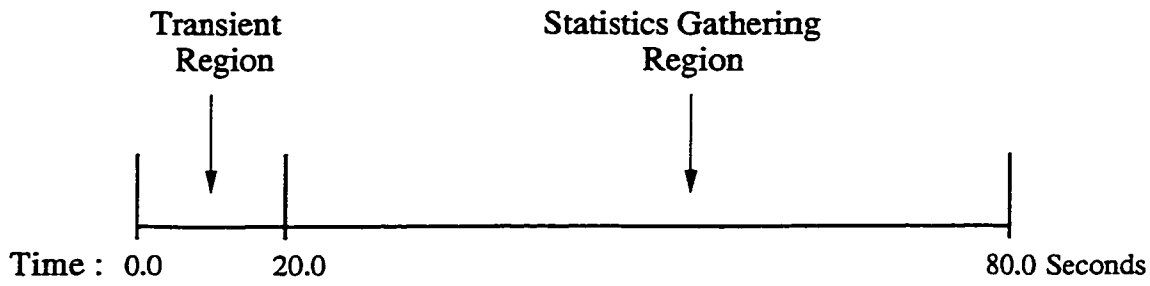


Figure 17: Two Regions of Simulation

4.4.1 Statistics Collection

We employ a simulation length of 80.0 seconds. The sources generate video data for the entire simulation length. This time can be split into two regions as in Figure 17. We name the two regions as: *transient region* and *statistics gathering* region. Since our aim is to capture the drop occurrences, we ignore the events during the first 20.0 seconds, which includes the time taken for the queue to fill up leading to drops. This can be illustrated in the following example. In an experiment employing shared buffers and shared bandwidth under synchronized frame arrivals, the first frame drop for 90% bandwidth case, occurs at time 5.17725 seconds; while, in the same experiment for 60% bandwidth case, we observe the first frame drop at 1.52919 seconds. On the other hand, experiments involving static bandwidth partition incur the first drop at around 18.0 seconds. Thus, we use statistics from the interval 20.0 to 80.0 seconds for generating the results. The statistics gathering length of 60.0 seconds is fixed for all experiments in this thesis. The results are quantified using the metrics defined in the next subsection.

4.4.2 Performance Measures

Traditionally, packets have been used in quantifying performance as they are the fundamental transmission units. But video data differs from this notion. A video frame typically consists of many packets. These packets together represent a picture complete with all the information. If packets belonging to a particular frame are lost, it will lead to an incomplete picture – affecting the quality. Thus, it is necessary to measure the performance of video streams in terms of frames, which can *loosely* correlate to *qualitative performance.*

We employ a measure called *Effective Frame Rate* (EFR). EFR can be defined as

the *rate of successfully sent frames.* It is measured over a one second quantum. In quantifying the performance of a *run* of a sequence, we use the average of EFRs over the *Simulation Length (SL)*; where, the *SL* corresponds to the statistics gathering region of 60 seconds. The Average EFR is computed as:

$$Average\ EFR = \frac{\sum_{time=1}^{SL} EFR_{time}}{SL}$$

Furthermore, in order to observe the extent of variation in fairness in an experiment, we employ *Fairness Index* (**FI**). Suppose, if EFR_1, EFR_2, ..., EFR_N represent Average EFRs for N sequences, then the FI is obtained by the standard deviation of all the Average EFRs as:

$$Variance\ of\ Average\ EFRs = \frac{\left(N * \sum_{seq=1}^{N} EFR_{seq}^2\right) - \left(\sum_{seq=1}^{N} EFR_{seq}\right)^2}{N * (N-1)}$$

$$Standard\ Deviation\ of\ Average\ EFRs\ (FI) = \sqrt{Variance\ of\ Average\ EFRs}$$

Apart from the fairness measure given by EFR, it is also necessary to know how the *quality fluctuates within a sequence over time.* Since the frame losses may not be uniform, we observe different EFRs over time. It is thus necessary to capture the variation in EFRs over the transmission time. This will enable us to know the extent of variation in quality when compared to the original sequence, which has a perfect frame rate of 30 fps. Thus, the variation of quality within a lossy sequence can be stated in terms of the deviation of EFR. Such a variation can occur in many ways. For example, if every other frame is sent, then we have a flat EFR curve with the deviation being zero. While in another case, if we have low EFRs in some intervals and high EFRs in some others, we have a fluctuating curve for EFR. We thus compute the standard deviation of EFR as:

$$Variance\ of\ EFR = \frac{\left(SL * \sum_{i=1}^{SL} EFR_i^2\right) - \left(\sum_{i=1}^{SL} EFR_i\right)^2}{SL * (SL-1)}$$

$$Standard\ Deviation\ of\ EFR = \sqrt{Variance\ of\ EFR}$$

We call the Average EFR for a run as simply "EFR" while specifying fairness in the following chapters. The deviation within a sequence represented by the Standard Deviation (SD) of EFR is denoted as "SD-of-EFR".

Video Name	average bandwidth *(ab)*	Buffer Size for Zero Loss at *ab*
bla	0.34344	1476
boo	0.55968	1035
can	0.62328	406
jet	0.59784	2695
mov	0.75048	915
tha	0.30528	1565
suk	0.54696	1080
sum	1.09392	3732
foo	1.18296	4652
boe	0.53424	2886

Table 5: Configuration Parameters for Effect of Bandwidth Variation

4.5 Effect of Bandwidth Variation on Performance

In this section, we present the effect of bandwidth variation on performance of each video sequence. For this purpose, we run each sequence independently with link capacity varied in steps of 5% over the entire range of average bandwidth of the respective sequences. The buffer size is fixed such that there is no loss for 100% of average bandwidth. Table 5 lists average bandwidth of each sequence and corresponding buffer size for the experiment. The frame arrivals are desynchronized, to reflect a realistic environment.

The performance in terms of EFR of each sequence is captured in Figures 18 through 27. We observe that the EFR decreases uniformly with the decrease in bandwidth, accompanied by negligible variations. Thus, the graphs exhibit almost linear plots, resulting in near monotonous drop in performance over the entire bandwidth region.

We capture the standard deviation of EFR of all the ten sequences in Table 6. The fluctuation is quite noticeable in some sequences as in case of `jet`, `tha`, `sum` and `foo`. Such a behaviour is attributed to the frame sizes consistently deviating away from their average frame size for long durations. On the contrary, sequences which fluctuate close to the average frame size result in low deviations of EFR. Examples of such sequences include `bla`, `can` and `mov`.

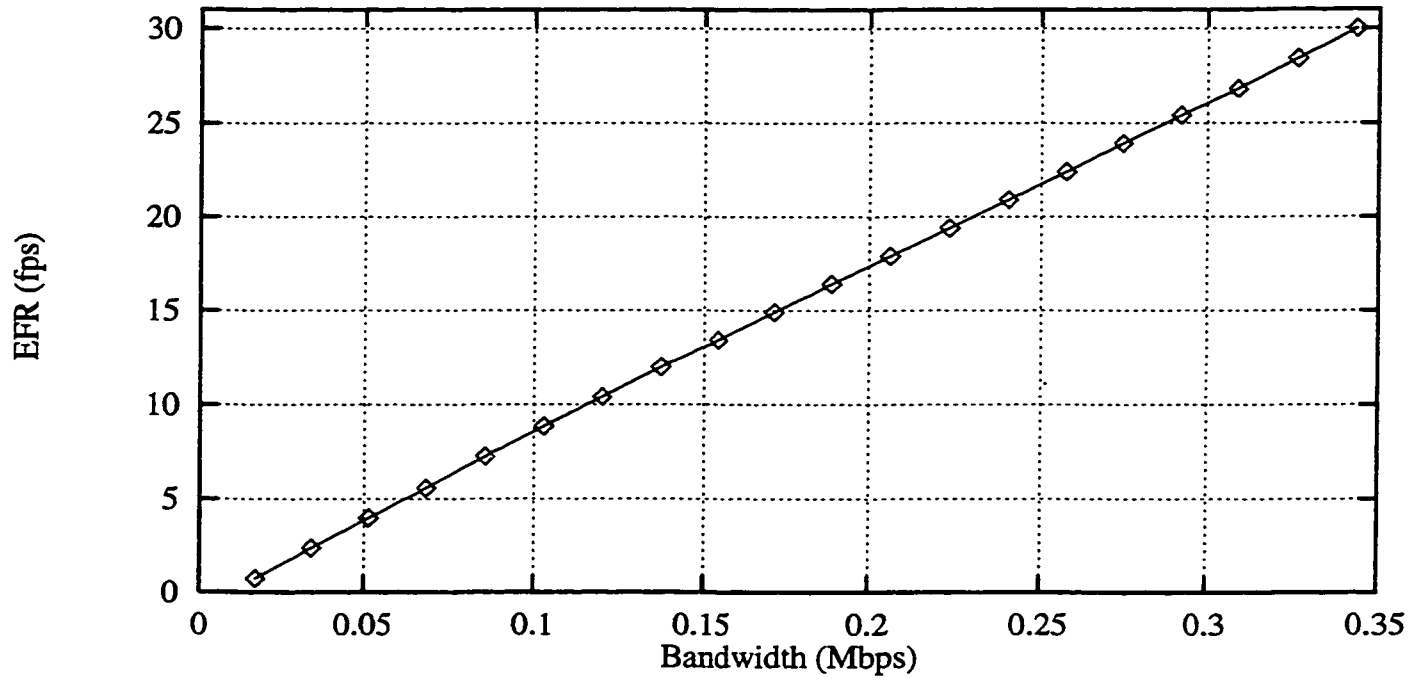


Figure 18: EFR of `bla` Sequence over Sample Bandwidth Points

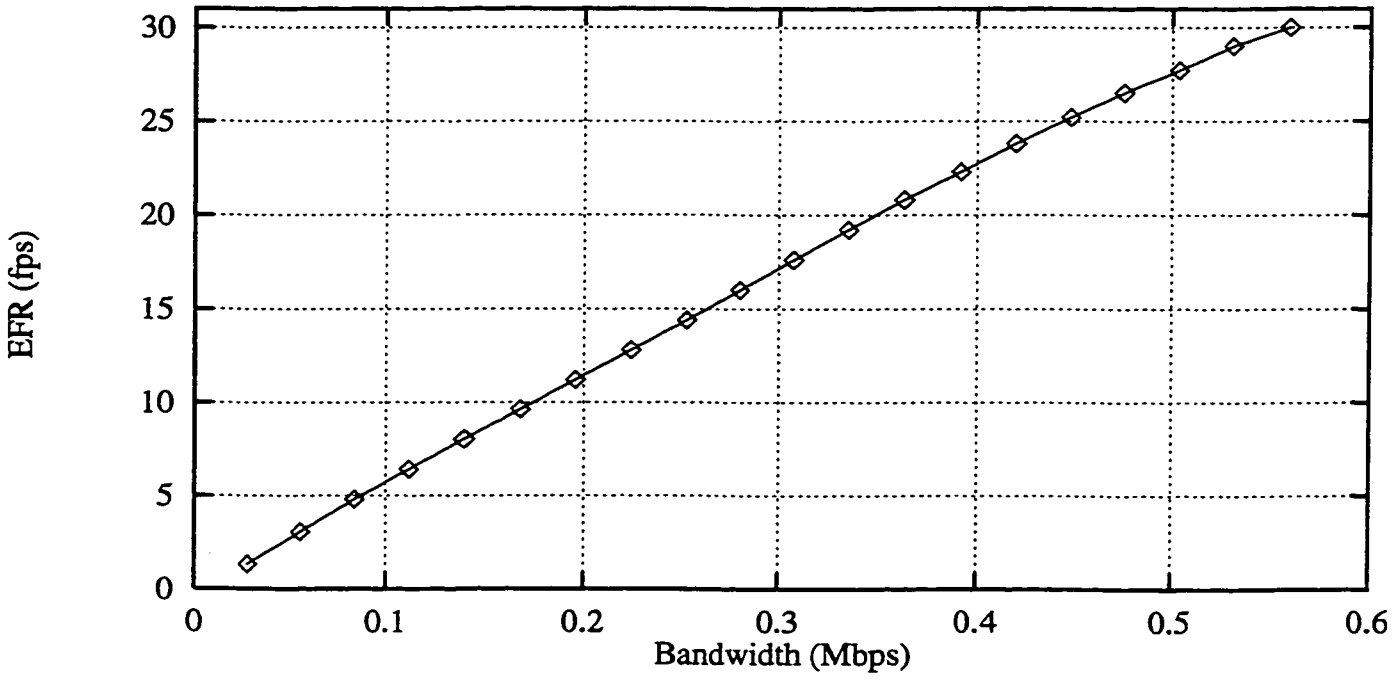


Figure 19: EFR of `boo` Sequence over Sample Bandwidth Points

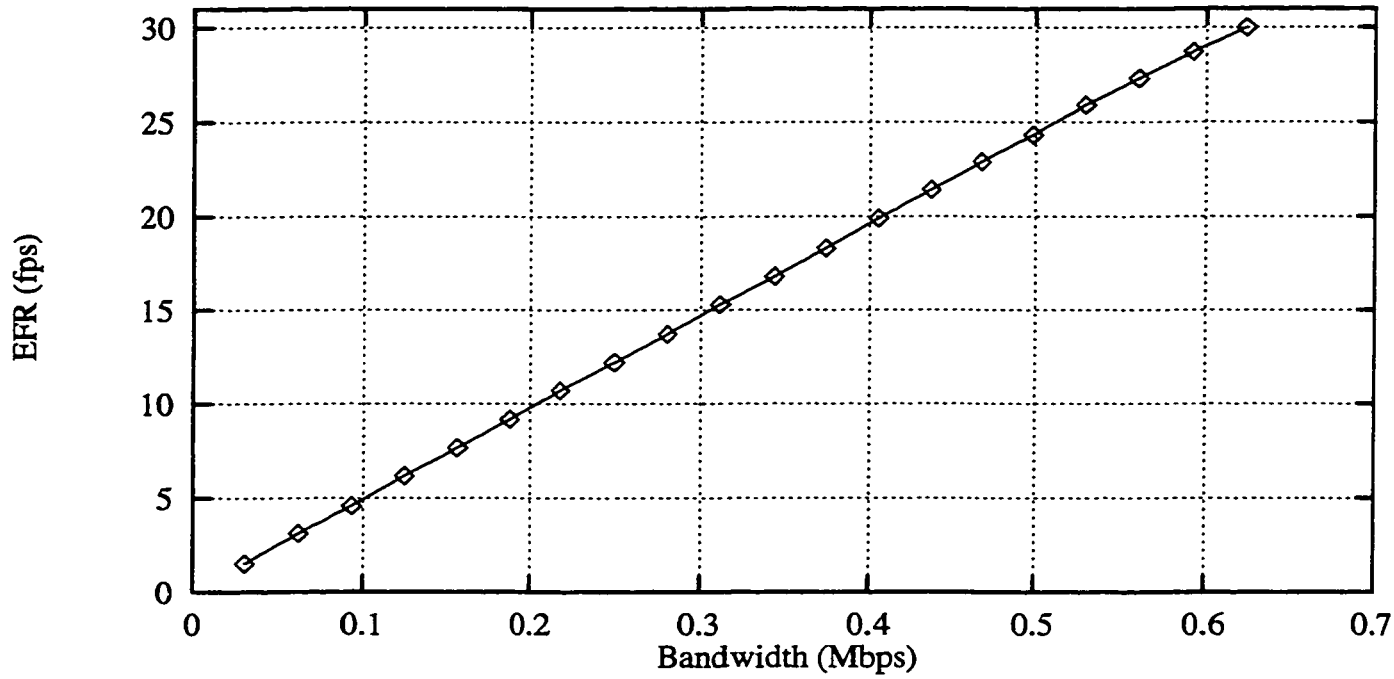


Figure 20: EFR of can Sequence over Sample Bandwidth Points

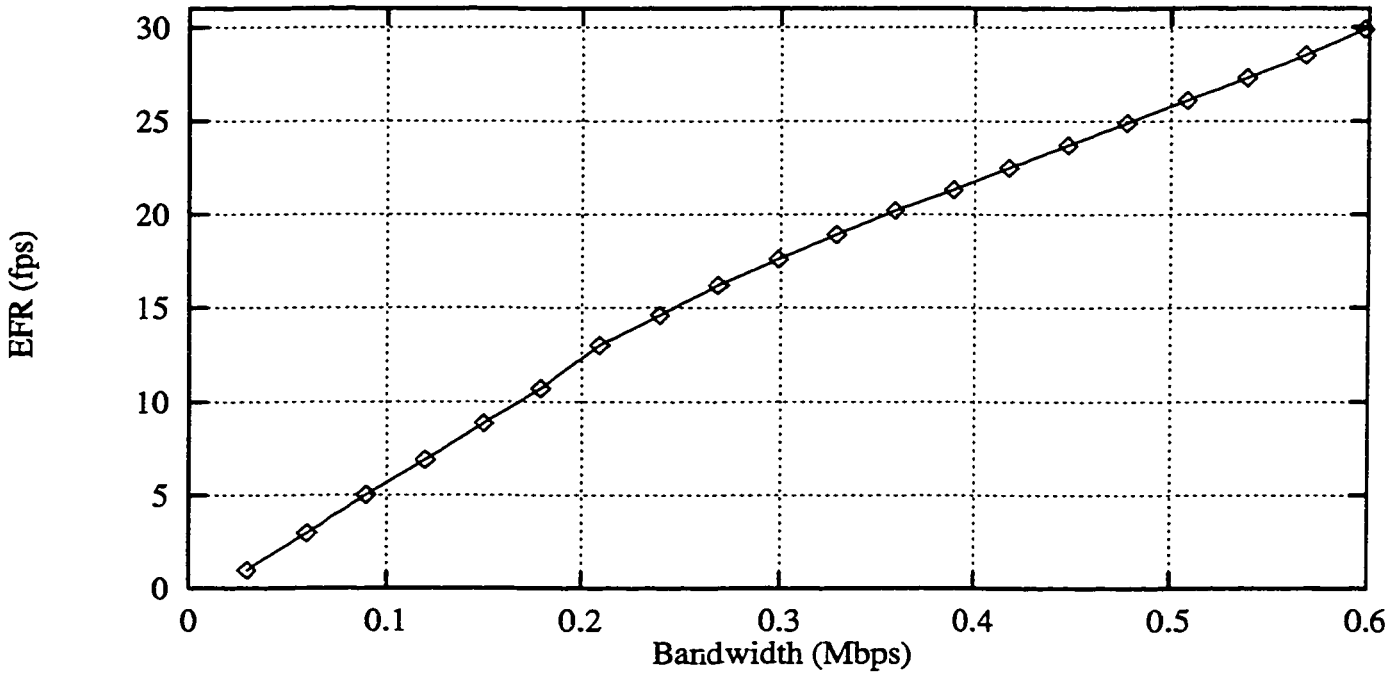


Figure 21: EFR of jet Sequence over Sample Bandwidth Points

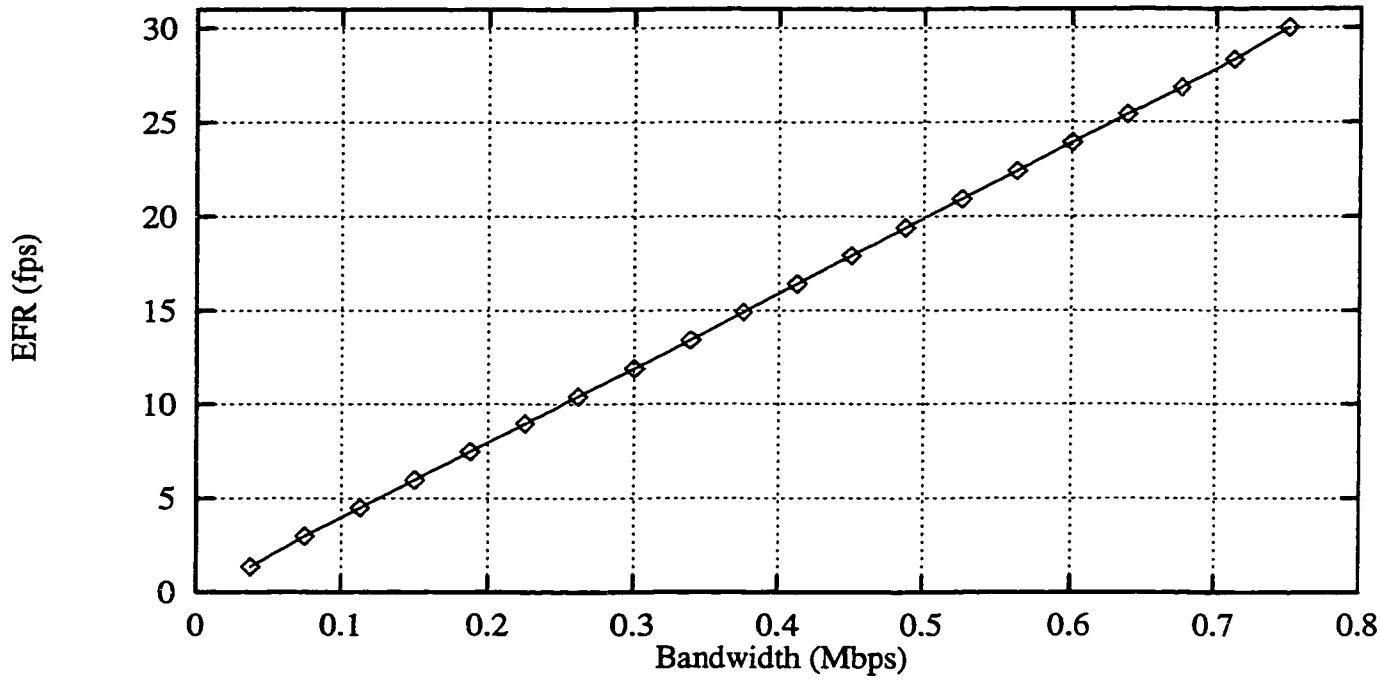


Figure 22: EFR of `mov` Sequence over Sample Bandwidth Points

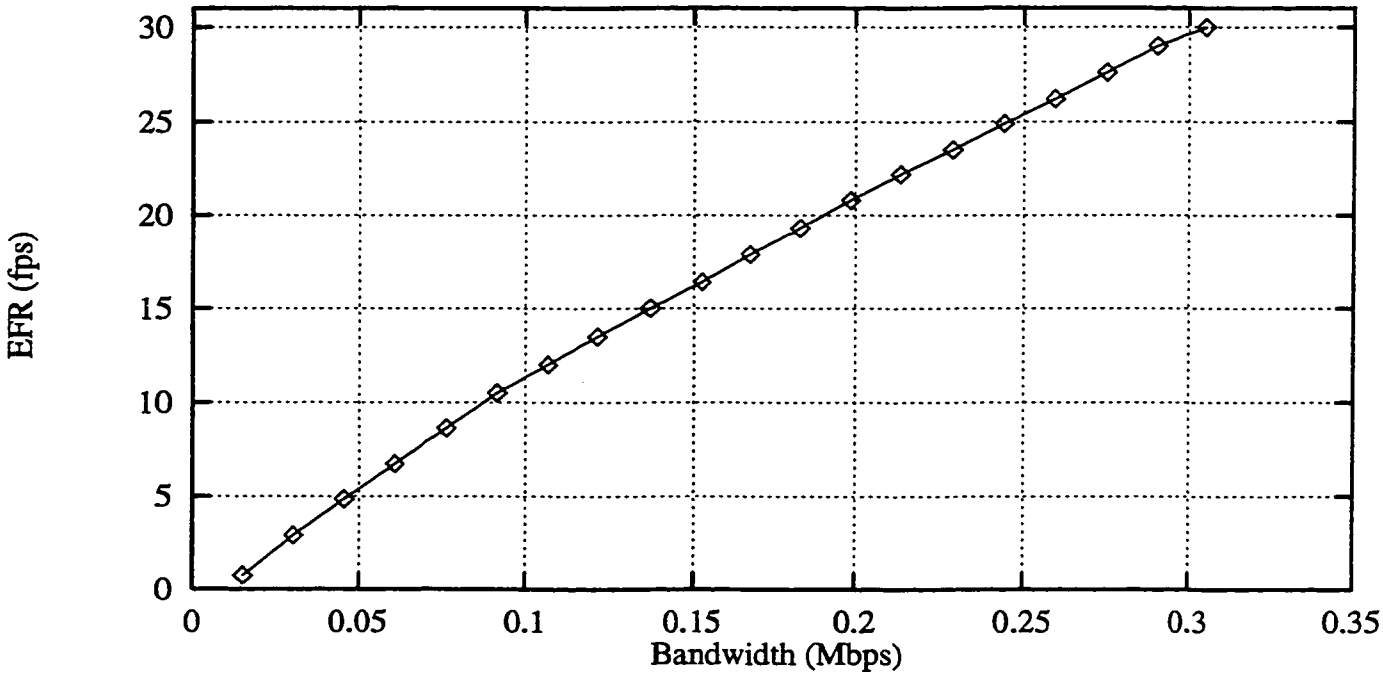


Figure 23: EFR of `tha` Sequence over Sample Bandwidth Points

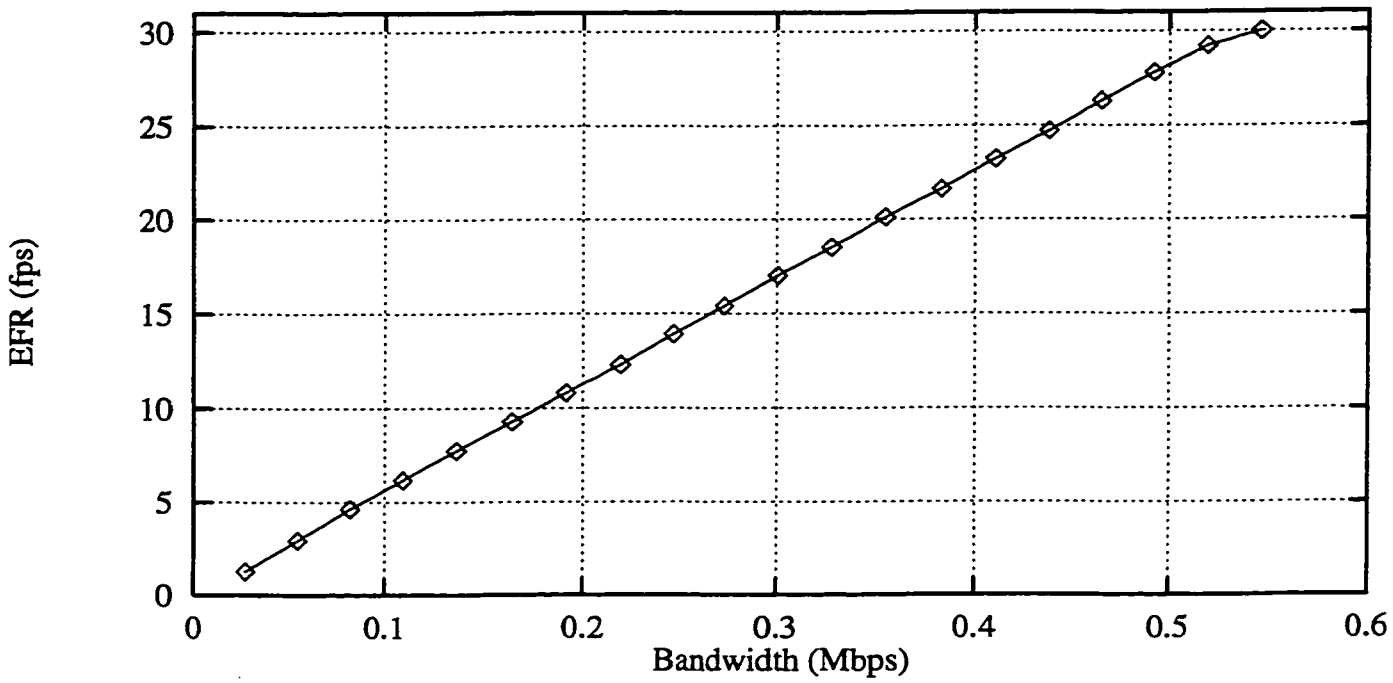


Figure 24: EFR of suk Sequence over Sample Bandwidth Points

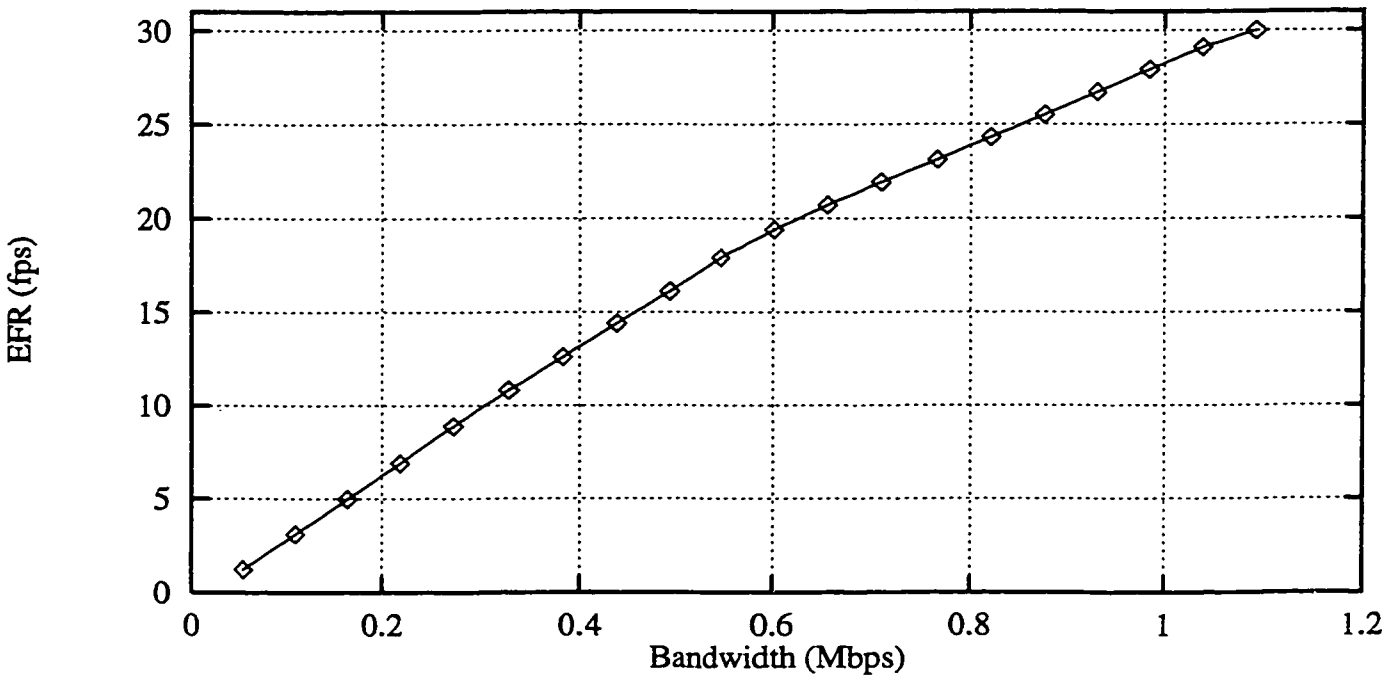


Figure 25: EFR of sum Sequence over Sample Bandwidth Points

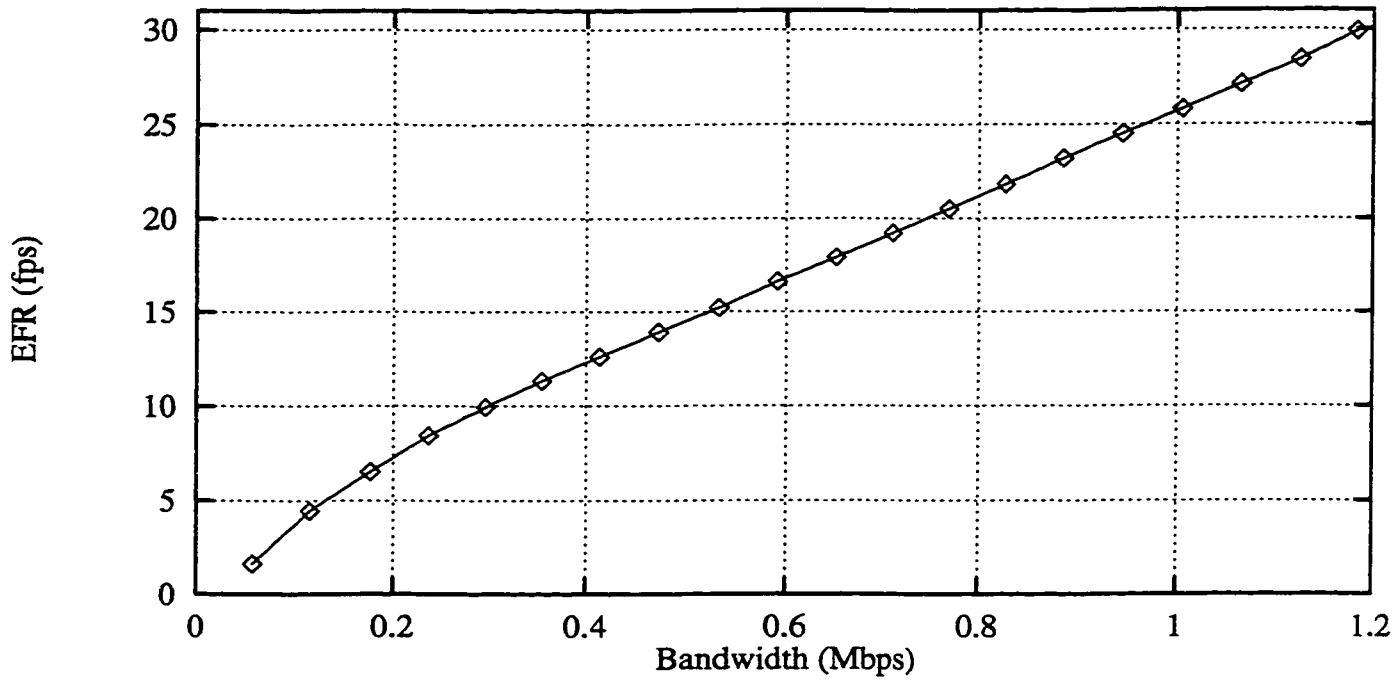


Figure 26: EFR of foo Sequence over Sample Bandwidth Points

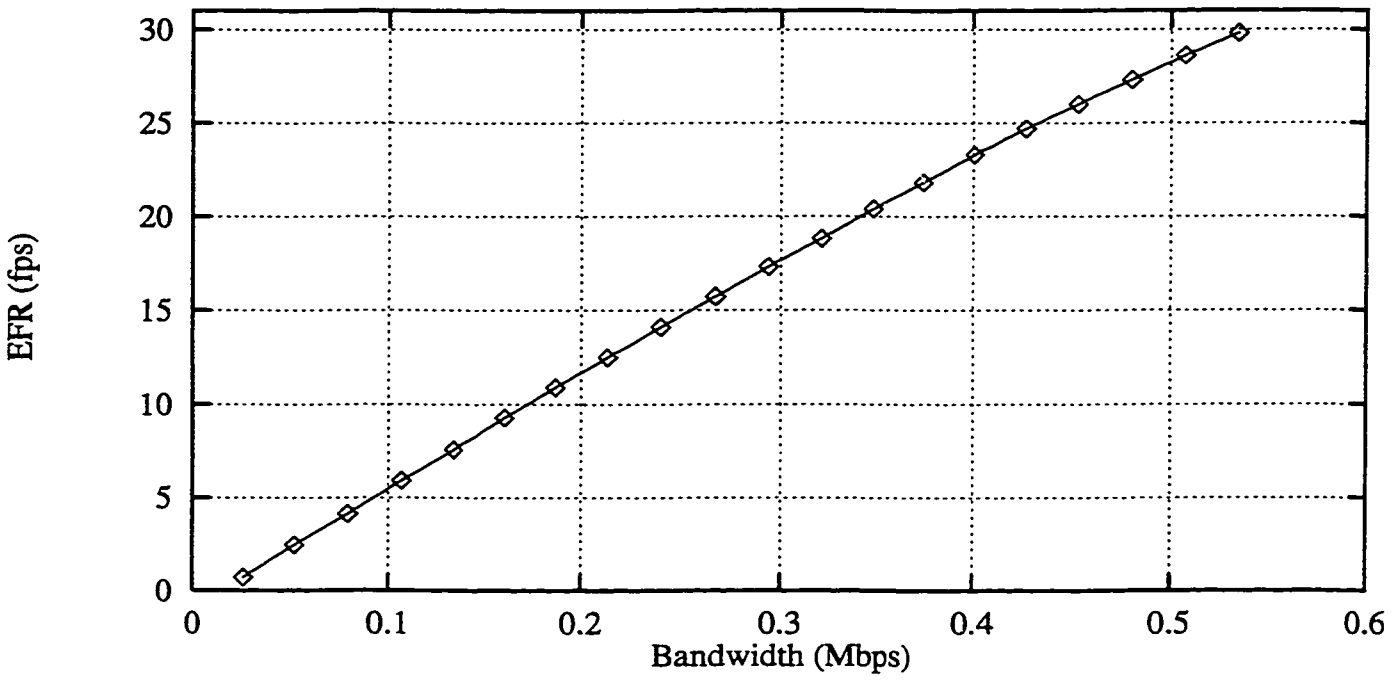


Figure 27: EFR of boe Sequence over Sample Bandwidth Points

	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
95%	2.26	2.1	1.36	1.63	0.52	1.81	1.68	1.92	1.93	2.52
90%	1.54	3.8	1.3	7.12	0.47	5.53	1.99	6.52	5.33	3.78
85%	1.31	3.33	1.23	7.29	0.53	5.79	1.74	5.96	5.45	3.32
80%	1.23	3.02	0.99	7.2	0.45	6.10	1.56	5.45	5.47	3.13
75%	1.1	2.61	0.86	6.41	0.5	6.28	1.37	4.77	5.57	3.16
70%	1.42	2.25	0.81	5.55	0.39	6.63	1.19	4.09	5.58	2.9
65%	1.62	1.9	0.68	4.94	0.5	6.07	0.99	3.4	6.04	2.74
60%	1.66	1.53	0.78	4.05	0.26	4.96	0.94	2.93	5.8	2.41
55%	1.56	1.2	0.52	3.25	0.5	3.9	0.8	2.4	5.9	2.07
50%	1.34	1.15	0.44	2.47	0.0	2.98	0.9	1.83	5.18	1.58
45%	0.86	0.8	0.5	1.3	0.63	1.32	0.75	1.21	2.42	0.85
40%	2.43	2.81	1.67	2.68	0.56	2.84	3.02	3.14	2.64	3.58
35%	2.49	3.81	1.75	3.49	0.56	3.35	3.13	4.09	3.37	3.95
30%	2.37	4.23	1.77	4.22	0.6	3.77	2.99	4.78	3.89	4.5
25%	2.18	4.55	1.7	4.93	0.5	4.39	2.8	5.39	4.21	4.79
20%	2.08	5.03	1.72	5.34	0.62	4.83	2.6	6.02	4.67	4.69
15%	1.99	4.86	1.63	5.88	0.5	5.1	2.44	6.27	5.05	4.62
10%	1.86	4.5	1.49	6.37	0.5	5.25	2.32	6.37	5.23	4.42
5%	1.67	4.17	1.36	6.74	0.5	5.45	2.12	6.54	5.38	4.05

Table 6: SD-of-EFR for Ten Video Streams Used in Simulations

Chapter 5

Shared Bandwidth and Shared Buffers

Our first set of experiments employ FCFS scheduling such that the bandwidth is completely shared between the competing streams. The buffers are also shared. When a new frame arrives at the gateway, the FAC checks for available space in the buffer pool and if space is available, the frame is queued in a single FIFO queue. If not, the frame is dropped. The enqueued frames are served in a FCFS manner. We start with FCFS scheduling due to its simplicity and least cost of implementation. We present additional experiments to show the effect of large frame sizes and varying buffer sizes on performance.

5.1 Overview of Experiments

We split the experiments into two categories: synchronized start and desynchronized frame arrivals. We distinguish between the two forms of arrivals by the desynchronization parameter. We achieve this by associating a value of 0 ms for synchronized arrivals and a non zero value representing the interval for desynchronized frame arrivals, which is 66 ms in our case, as described in Chapter 4.

The configuration parameters are given in Table 7. Buffer size is selected such that there are no losses for 100% of Average Bandwidth, which is the sum of average bandwidths of all ten streams. We gather statistics over a period of 60 seconds, as outlined in Chapter 4. The link capacity is varied in steps of 10% in the range [100%,

Configuration Parameter	Value
Number of Sources	10
Buffer Size	8046 cells
Link Bandwidth	6.54 - 3.92 Mbps
Frame Desynchronization Period	00 or 66 ms

Table 7: Configuration Parameters for Shared Bandwidth and Shared Buffers

60%] of Average Bandwidth, to reflect the constrained bandwidth region. Apart from the frame desynchronization period, other parameters do not change unless otherwise explicitly stated in an experiment.

We employ three runs for each form of arrivals, wherein every run employs a different set of randomly ordered video sequences. These are called Order1, Order2 and Order3. Each set consists of ten video sequences corresponding to the ten sources in our simulation environment. These sequences are mapped to the corresponding sources before the start of simulation run. Actual mapping for the three different orders is given in Table 8.

Source	Sequence
0	bla
1	boo
2	can
3	jet
4	mov
5	tha
6	suk
7	sum
8	foo
9	boe

Source	Sequence
0	bla
1	sum
2	tha
3	suk
4	foo
5	can
6	jet
7	boe
8	boo
9	mov

Source	Sequence
0	bla
1	foo
2	boe
3	tha
4	boo
5	jet
6	can
7	suk
8	sum
9	mov

Table 8: Mapping for Order1 (left), Order2 (middle) and Order3 (right)

5.2 Results

In this section, we present experiments to capture the effect of sharing both bandwidth and buffers under synchronized and desynchronized frame arrivals.

5.2.1 Synchronized Start

In synchronized arrivals, all the ten participating sequences start at the same time. As a result, frames corresponding to the ten sequences arrive at their respective arrival times, based on the frame rate of 30 fps. We deactivate desynchronization parameter with a value of 0 ms in this experiment.

The EFR depicted in Table 9 for the ten sequences exhibits typical FCFS behaviour. A prime observation is that, sequences in the beginning of an order almost always succeed in sending their frames, while the sequences at the rear end suffer from losses. As expected, the number of sequences experiencing losses increase as bandwidth is reduced to lower levels. For instance, in orders 1 and 3, for 90% bandwidth case, only two sequences suffered losses, whereas in 60% case four or more sequences incurred drops.

Another prime observation appears in the form of high unfairness in loss distribution. For example, in Order1, for 60% bandwidth, the EFR range achieved is 30.0 to 2.7 fps. This is reflected by the higher values of Fairness Index (FI).

The variation of EFRs is presented in Table 10. The SD-of-EFR shows no distinct patterns for the three orders. Drop occurrences in the sequences fluctuate very widely. This can be illustrated as follows: in Order2, for 80% bandwidth, the sequence `boo` has a SD of 9.42; while, it is 3.77 for 60% bandwidth. A still worse case in Order2 is the `mov` sequence where the SD jumps from 0.18 to 10.5, for 60% to 90% bandwidth. This behaviour coupled with the unfairness results in a high degree of adverse effect on visual quality. As sources successful in sending all their frames will result in perfect quality, the others incurring losses yield poor quality.

5.2.2 Frame Arrival Desynchronization

Desynchronization of frame arrivals is implemented by randomly varying the frame arrival times over the desynchronization period. In this experiment we activate desynchronization parameter with a value of 66 ms. As a result, two consecutive frames

Order1	Video Sequences										**FI**
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.0
90%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	19.3	20.9	4.19
80%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	28.0	8.5	15.9	7.63
70%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	17.0	3.9	16.9	9.12
60%	30.0	30.0	30.0	30.0	30.0	30.0	26.7	4.0	2.7	9.3	11.8

Order2	Video Sequences										**FI**
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.0
90%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	29.1	22.1	11.0	6.21
80%	30.0	30.0	30.0	30.0	30.0	30.0	27.2	20.9	10.9	2.4	9.86
70%	30.0	30.0	30.0	30.0	30.0	27.2	16.3	10.3	4.8	0.4	11.8
60%	30.0	30.0	30.0	30.0	27.1	17.4	8.2	5.4	2.7	0.0	12.7

Order3	Video Sequences										**FI**
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.0
90%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	22.3	18.0	4.28
80%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	29.7	10.1	9.8	8.44
70%	30.0	30.0	30.0	30.0	30.0	30.0	28.2	18.2	2.3	2.1	11.6
60%	30.0	30.0	30.0	30.0	30.0	27.3	14.0	7.7	0.5	0.5	12.9

Table 9: EFR of Synchronized Start for Shared Bandwidth and Shared Buffers

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.60	5.45
80 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.44	8.22	5.73
70 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.78	5.62	5.54
60 %	0.00	0.00	0.00	0.00	0.00	0.00	5.29	6.63	5.15	7.76

Order2	Video Sequences									
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.99	8.05	10.5
80 %	0.00	0.00	0.00	0.00	0.00	0.00	5.39	7.76	9.42	5.32
70 %	0.00	0.00	0.00	0.00	0.00	5.59	10.0	8.29	5.61	1.53
60 %	0.00	0.00	0.00	0.00	4.82	10.2	8.17	4.84	3.77	0.18

Order3	Video Sequences									
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.87	7.67
80 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.73	9.36	7.49
70 %	0.00	0.00	0.00	0.00	0.00	0.00	3.25	7.23	5.00	3.64
60 %	0.00	0.00	0.00	0.00	0.00	4.86	7.77	6.23	1.62	1.47

Table 10: SD-of-EFR of Synchronized Start for Shared Bandwidth and Shared Buffers

Order1	Video Sequences										FI
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.0
90%	29.5	28.3	28.5	28.4	27.9	29.4	28.9	25.2	25.0	28.7	1.59
80%	28.6	26.8	26.9	26.6	25.8	29.2	27.5	21.2	19.1	27.3	3.22
70%	27.9	25.4	25.0	24.6	22.7	28.3	25.8	16.9	14.0	26.1	4.66
60%	27.2	23.2	22.5	22.3	19.5	28.0	24.2	12.7	10.0	24.2	5.84

Order2	Video Sequences										FI
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.0
90%	29.4	25.0	29.6	28.9	25.0	28.6	28.3	28.8	28.7	27.8	1.66
80%	28.8	20.8	29.1	27.4	19.1	27.0	26.5	27.5	27.1	25.9	3.31
70%	28.3	16.9	28.6	26.0	14.0	24.6	24.9	25.7	25.1	23.2	4.71
60%	27.7	12.7	27.7	23.9	10.0	22.6	21.9	23.9	23.6	19.9	5.83

Order3	Video Sequences										FI
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.0
90%	29.6	24.2	29.0	29.8	28.8	28.3	28.5	28.8	25.6	28.0	1.78
80%	28.9	18.6	27.5	29.2	27.4	26.9	26.7	27.2	21.5	25.7	3.34
70%	28.3	13.9	26.1	28.3	25.5	24.3	24.4	25.6	17.3	23.3	4.63
60%	27.2	9.6	23.9	28.0	23.4	22.2	22.1	23.8	13.3	20.3	5.78

Table 11: EFR of Desynchronized Frame Arrivals for Shared Bandwidth and Shared Buffers

may arrive with a maximum gap of roughly twice the inter-arrival time. On the other hand, frames can arrive with a much shorter gap.

Frame level desynchronization shows considerable improvement over the synchronized arrivals as shown in Table 11. We observe that higher bandwidth values in the constrained region show better results. For example, in Order1 for 90% bandwidth case, the range of EFRs achieved is 29.5 to 25.0 fps. Similar trend continues for the other two orders. On the other hand, as the bandwidth drops toward 60% mark, the unevenness in EFR becomes quite visible. For instance, the best and worst EFRs of 60% bandwidth case, for orders 1, 2 and 3 are: 28.0 and 10.0 fps; 27.7 and 10.0 fps; 28.0 and 9.6 fps respectively. Such a variation in fairness captured by the FI shows considerable improvement over that of synchronized arrivals.

We notice that tha achieves the highest individual EFR in all the three orders,

which has the least average frame size. On the contrary, `foo` records the lowest EFR, which has the second highest average frame size in the set of ten sequences and highest deviation of frame sizes. We also notice that `foo` and `sum` achieve the lowest EFR in all the three orders when compared to other sequences. We find from the frame size characteristics that these two streams have larger frame sizes among all the sequences.

The variation in EFR depicted in Table 12, also exhibits major improvements compared to the synchronized start, where the SD-of-EFR was around 10 in few instances. Low SD-of-EFR is achieved by sequences that are statistically less variant. Examples of such sequences include `bla` and `tha` in all the three orders. On the contrary, sequences which are statistically more variant exhibit higher variation. For instance, `sum` in all the three orders. As in the case of fairness, desynchronized arrivals lead to considerable improvements in the outcome compared to synchronized arrivals.

5.3 Summary

We present the highlights of sharing both bandwidth and buffers below:

- In terms of *fairness*, synchronized arrivals case fares poorly whereas frame level desynchronization leads to considerable improvement. However, the fairness is not very good especially when bandwidth is more constrained. In addition, sequences with large frames are unfairly treated when compared to other streams.
- In terms of *variation within a sequence*, synchronized start exhibits higher values. The situation improves towards a better note in frame level desynchronization. But, there is still potential for further improvement.

5.4 Additional Observations

In this section, we include additional experiments to illustrate the effect of FCFS scheduling technique on sequences with large frame sizes and the influence of buffer size variations on performance.

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90 %	0.72	1.61	1.41	1.66	1.78	0.95	1.21	3.55	3.37	1.38
80 %	1.14	2.24	1.80	2.43	2.29	0.95	1.72	5.00	3.94	1.53
70 %	1.38	2.87	2.46	3.13	2.27	1.39	2.61	5.70	4.63	2.49
60 %	1.61	3.88	2.81	4.12	2.59	1.90	2.46	6.62	5.02	3.22

Order2	Video Sequences									
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90 %	0.89	3.76	0.57	1.26	3.28	1.19	1.81	1.44	1.21	1.72
80 %	1.15	4.83	0.96	1.72	4.29	1.30	2.75	1.76	2.24	2.05
70 %	1.39	5.91	1.22	2.06	4.62	1.93	3.33	2.43	2.46	2.21
60 %	1.65	6.39	1.61	2.66	5.86	2.78	4.21	3.09	3.40	2.94

Order3	Video Sequences									
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90 %	0.76	3.83	1.14	0.51	1.49	1.84	1.51	1.33	3.37	1.48
80 %	1.07	4.18	1.73	0.85	1.83	2.48	1.80	2.07	4.82	2.19
70 %	1.39	4.97	2.39	1.38	2.93	3.81	1.75	2.20	5.36	2.47
60 %	1.72	5.79	3.33	1.48	3.59	4.05	2.61	2.17	6.22	2.71

Table 12: SD-of-EFR of Desynchronized Frame Arrivals for Shared Bandwidth and Shared Buffers

Configuration Parameter	Pair1	Pair2
Buffer Size	4600 cells	5772 cells
Link Bandwidth	1.4 - 0.84 Mbps	1.53 - 0.92 Mbps

Table 13: Configuration Parameters for Frame Size Effect

5.4.1 Frame Size Effect

Previous experiment shows unfair treatment for streams with large frame sizes. In order to validate this effect, the current test is conducted. The idea behind the present experiment is to determine whether the stream with larger frame sizes always performs lower than other streams. For this purpose, we have selected two pairs of sequences based on their frame size characteristics. Each pair consists of two sequences, one with smaller frame sizes and another with larger frame sizes. The two pairs are (`tha`, `sum`) (Pair1) and (`bla`, `foo`) (Pair2). Among these sequences, `sum` and `foo` are larger ones, while `tha` and `bla` are the smaller ones.

Most of the configuration parameters are modified as only two sequences are used in this experiment. The variable parameters are listed in Table 13 which includes bandwidth and buffer size for the two pairs of sequences considered. Bandwidth is the sum of the average bandwidth of two streams. The buffer size is set such that there are no losses for either sequences at their combined average bandwidth.

We conducted two runs for each pair, wherein the position of sequences was interchanged in each run. The outcome is presented in Tables 14 and 15. The smaller sequences `tha` and `bla` achieve a higher EFR in both the positions for all the bandwidth points. On the other hand, the larger sequences `sum` and `foo` suffer considerable losses. For instance, `tha` achieves an EFR of 28.9 fps, while `sum` records around 17.1 fps for 60% bandwidth case. This, indicates that FIFO offers unfair treatment to sequences with larger frame sizes.

5.4.2 Buffer Size Effect

In this section, we consider the effect of buffer size on the performance measures. We conducted several experiments to determine the effect of variations in buffer size over the performance of ten sequences. Here, we present one such instance of an

	Video Sequences	
Bandwidth	tha	sum
100 %	30.0	30.0
90 %	29.8	27.4
80 %	29.5	24.4
70 %	29.3	21.4
60 %	28.6	17.9

	Video Sequences	
Bandwidth	sum	tha
100 %	30.0	30.0
90 %	27.4	29.7
80 %	24.4	29.4
70 %	21.4	29.3
60 %	17.1	29.0

Table 14: Frame Size Effect on Fairness for tha and sum

	Video Sequences	
Bandwidth	bla	foo
100 %	30.0	30.0
90 %	29.7	26.3
80 %	29.2	23.0
70 %	29.1	19.6
60 %	28.6	16.4

	Video Sequences	
Bandwidth	foo	bla
100 %	28.6	29.2
90 %	24.7	28.5
80 %	21.4	28.0
70 %	18.2	27.8
60 %	15.0	27.1

Table 15: Frame Size Effect on Fairness for bla and foo

experiment for Order1. The configuration parameters are similar to the ones employed in desynchronized frame arrivals case in the previous section. Notable changes are: the bandwidth which is fixed at 90% of Average Bandwidth (5.88 Mbps) and the buffer size which is varied in steps of 100, starting from 8046 cells till 8446 cells.

Table 16 depicts the EFR and its variation. The first column lists the buffer sizes instead of bandwidth. The EFR values indicate no variations. We observe similar outcome in terms of SD-of-EFR. Thus, we conclude that buffer size has no effect on performance, when the link capacity and associated buffers are shared.

Order1	Video Sequences									
Buffer Size	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
8046	29.5	28.3	28.5	28.4	27.9	29.4	28.9	25.2	25.0	28.7
8146	29.5	28.3	28.5	28.4	27.9	29.4	28.9	25.2	25.0	28.7
8246	29.5	28.3	28.5	28.4	27.9	29.4	28.9	25.2	25.0	28.7
8346	29.5	28.3	28.5	28.4	27.9	29.4	28.9	25.2	25.0	28.7
8446	29.5	28.3	28.5	28.4	27.9	29.4	28.9	25.2	25.0	28.7

Order1	Video Sequences									
Buffer Size	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
8046	0.72	1.61	1.41	1.66	1.78	0.95	1.21	3.55	3.37	1.38
8146	0.72	1.61	1.41	1.66	1.78	0.95	1.21	3.55	3.37	1.38
8246	0.72	1.61	1.41	1.66	1.78	0.95	1.21	3.55	3.37	1.38
8346	0.72	1.61	1.41	1.66	1.78	0.95	1.21	3.55	3.37	1.38
8446	0.72	1.61	1.41	1.66	1.78	0.95	1.21	3.55	3.37	1.38

Table 16: Buffer Size Effect on EFR (top) and SD-of-EFR (bottom)

Chapter 6

Static Buffer Partition

Shared bandwidth and buffers resulted in high unfairness, wherein some streams experienced large drops. We also noticed higher deviations in quality within the sequences. In order to improve the performance, we need to provide an "equal" opportunity for each stream to get their frames through the FAC. Such a requirement points at partitioning the resources. Thus, the idea is to split the buffers once for the entire duration of transmission. We anticipate a better spread of drops across the streams, with such a partition which could lead to an improved visual quality video at the sink.

In the current set of experiments, we employ FCFS scheduling technique so that the link capacity is shared among the participating streams. On the other hand, buffers are partitioned and thereby each sequence has an independent pool of buffers dedicated to it. Upon arrival of a new frame, the FAC tests for the availability of sufficient buffers in the independent FIFO buffer space and if found, the frame is queued. Otherwise, it is dropped. The enqueued frames are served in an FCFS basis. We use FCFS scheduler because of its simple implementation and least cost.

6.1 Buffer Partitioning

We consider partitioning the available buffers *equally* among the ten sequences. For partitioning, we use the buffer size employed in Chapter 5. A shared buffer size of 8050 cells was sufficient to send all ten sequences without any loss at 100% of *Average Bandwidth*. Equally partitioning the buffer space of 8050 cells results in 805 cells per

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	29.5	29.4	30.0
90%	30.0	29.9	30.0	29.8	29.9	30.0	30.0	23.9	21.9	29.9
80%	30.0	28.6	29.1	27.6	25.9	30.0	29.6	19.8	16.5	29.2
70%	30.0	26.4	25.0	24.4	20.8	30.0	28.0	16.5	13.9	27.1
60%	29.9	23.2	20.6	21.2	16.7	29.9	23.9	13.4	12.0	23.7

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	1.26	0.00
90 %	0.00	0.28	0.00	0.56	0.29	0.00	0.00	4.73	3.35	0.18
80 %	0.00	1.83	1.22	2.77	1.44	0.00	0.80	6.27	4.08	1.66
70 %	0.00	3.44	1.99	4.01	1.31	0.00	2.63	6.12	4.72	3.18
60 %	0.40	4.89	1.90	5.47	1.18	0.29	3.07	5.19	4.96	4.20

Table 17: EFR (top) and SD-of-EFR (bottom) for Equally Partitioned Large Buffers

source. Applying this value we conduct the first experiment. In order to determine the effect of such a partition, in the next experiment we reduce the buffer size for each sequence by half to 403 cells. Apart from these independent buffer sizes, other configuration parameters are identical to the ones declared in the previous Chapter. We employ desynchronized frame arrivals to reflect a realistic environment.

The results of small and large buffers for equal split show similar behaviour. Both these cases show negligible variations in terms of fairness. A similar trend is observed in the case of deviation within a sequence aspect. We present EFR and its deviation in Table 17. In order to determine the effect of unequal partitioning, we conducted other experiments. The results of these experiments are not presented here as its buffer operating region is not in our domain[1]. Deriving from these tests we found that till all the sources get a fair number of buffers, the performance does not deteriorate. In other words, the split buffers should not differ from one another in large magnitudes. Based on these observations we opt for equal buffer split of larger quantity.

[1]We do not consider the issue of constrained buffer space in this thesis.

6.2 Overview of Experiments

We conduct experiments under synchronized and desynchronized frame arrivals, distinguished by the desynchronization parameter. We maintain the bandwidth as a shared resource. On the other hand, buffer usage receives a change in the form of static partitioning where, each stream has an independent buffer space of 805 cells. For each form of frame arrivals, we make use of the three orders described in the last Chapter.

6.3 Results

In this section, we describe experiments to examine the effect of sharing link bandwidth with equally partitioned buffers as outlined in the previous section.

6.3.1 Synchronized Start

In the synchronized arrival case, frames corresponding to the ten sources arrive roughly at their respective arrival times which are spaced apart by an inter-arrival time of 33 ms. We employ the parameters mentioned in the overview section with a value of 0 ms for frame desynchronization parameter.

The synchronized start shows a sizable improvement in performance over similar experiment in Chapter 5. The frame drops, as expected are better spread out and are depicted in Table 18. A key observation is that, every sequence in a different position, according to the three orders performs in an identical manner with negligible difference. As regards fairness, though we notice improvements compared to the synchronized arrival case of shared bandwidth and shared buffers, there is still considerable disparity. For instance, in Order2, 60% bandwidth case results in an EFR range of 29.9 to 12.0 fps. The variation in fairness is still high. Thus, there is a potential for further improvement in performance.

Variation within a sequence also shows a downward trend, as can be seen in Table 19. Sequences with high resource demands suffer from large variations. Examples of such a behaviour include `foo`, `jet` and `sum`. In fact, these sequences are at the higher end of frame size deviation among the ten sequences.

Order1	Video Sequences										**FI**
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	29.6	29.4	30.0	0.22
90%	30.0	30.0	30.0	29.9	30.0	30.0	30.0	23.9	21.8	29.9	3.04
80%	30.0	28.8	29.4	27.9	26.1	30.0	29.6	19.7	16.3	29.1	4.79
70%	30.0	26.7	25.5	24.5	20.8	30.0	28.1	16.4	13.7	26.9	5.58
60%	29.9	23.5	20.8	21.3	16.8	29.9	23.8	13.3	12.0	23.5	6.1

Order2	Video Sequences										**FI**
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov	
100%	30.0	29.6	30.0	30.0	29.4	30.0	30.0	30.0	30.0	30.0	0.22
90%	30.0	24.1	30.0	30.0	21.8	30.0	29.8	29.9	29.8	29.9	2.99
80%	30.0	20.2	30.0	29.7	16.7	29.2	27.6	29.1	28.5	25.4	4.6
70%	30.0	16.8	30.0	28.2	14.0	25.2	24.3	27.1	26.3	20.4	5.48
60%	29.9	13.6	29.9	24.2	12.0	20.6	21.2	23.6	23.1	16.5	6.09

Order3	Video Sequences										**FI**
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov	
100%	30.0	29.6	30.0	30.0	30.0	30.0	30.0	30.0	29.4	30.0	0.22
90%	30.0	22.3	29.9	30.0	29.9	29.9	30.0	30.0	23.5	29.9	2.99
80%	30.0	16.9	29.3	30.0	28.7	27.7	29.2	29.6	19.6	25.5	4.67
70%	30.0	14.2	27.4	30.0	26.6	24.4	25.0	28.0	16.3	20.4	5.53
60%	29.9	12.2	24.0	29.9	23.3	21.3	20.5	23.8	13.2	16.5	6.12

Table 18: EFR of Synchronized Start for Statically Partitioned Buffers

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.20	1.45	0.00
90 %	0.00	0.00	0.00	0.26	0.00	0.00	0.00	4.62	3.50	0.18
80 %	0.00	1.93	0.93	2.56	1.56	0.00	0.82	6.27	4.10	1.73
70 %	0.00	3.22	2.10	3.88	1.59	0.00	2.62	6.16	4.65	3.33
60 %	0.22	4.81	1.96	5.39	1.13	0.29	3.07	5.27	5.15	4.20

Order2	Video Sequences									
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov
100 %	0.00	1.07	0.00	0.00	1.27	0.00	0.00	0.00	0.00	0.00
90 %	0.00	4.56	0.00	0.00	3.53	0.00	0.55	0.18	0.37	0.33
80 %	0.00	6.13	0.00	0.65	4.04	1.08	2.67	1.67	1.88	1.64
70 %	0.00	6.21	0.00	2.57	4.70	2.03	3.91	3.30	3.47	1.65
60 %	0.18	5.37	0.26	3.15	5.05	1.85	5.39	4.15	4.92	1.03

Order3	Video Sequences									
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov
100 %	0.00	0.86	0.00	0.00	0.00	0.00	0.00	0.00	1.47	0.00
90 %	0.00	3.41	0.18	0.00	0.18	0.38	0.00	0.00	4.88	0.33
80 %	0.00	4.08	1.56	0.00	1.95	2.69	1.12	0.80	6.32	1.60
70 %	0.00	4.67	3.06	0.00	3.39	3.90	2.05	2.67	6.16	1.74
60 %	0.22	4.95	4.30	0.26	4.66	5.40	2.05	3.04	5.20	0.96

Table 19: SD-of-EFR of Synchronized Start for Statically Partitioned Buffers

6.3.2 Frame Arrival Desynchronization

In this experiment, we examine the effect of desynchronized frame arrivals with statically split buffers and shared bandwidth, on performance. We employ the same set of parameters that were used in the previous experiment with frame level desynchronization activated.

Order1	Video Sequences										**FI**
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	29.5	29.4	30.0	0.23
90%	30.0	29.9	30.0	29.8	29.9	30.0	30.0	23.9	21.9	29.9	3.01
80%	30.0	28.6	29.1	27.6	25.9	30.0	29.6	19.8	16.5	29.2	4.7
70%	30.0	26.4	25.0	24.4	20.8	30.0	28.0	16.5	13.9	27.1	5.5
60%	29.9	23.2	20.6	21.2	16.7	29.9	23.9	13.4	12.0	23.7	6.1

Order2	Video Sequences										**FI**
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov	
100%	30.0	29.5	30.0	30.0	29.4	30.0	30.0	30.0	30.0	30.0	0.23
90%	30.0	23.8	30.0	30.0	21.9	30.0	29.9	29.9	29.9	29.9	3.03
80%	30.0	19.8	30.0	29.6	16.6	29.1	27.6	29.2	28.6	25.9	4.68
70%	30.0	16.5	30.0	28.1	13.9	25.1	24.4	27.1	26.4	20.7	5.52
60%	29.9	13.4	29.9	23.8	12.1	20.6	21.3	23.7	23.2	16.7	6.07

Order3	Video Sequences										**FI**
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov	
100%	30.0	29.4	30.0	30.0	30.0	30.0	30.0	30.0	29.5	30.0	0.23
90%	30.0	21.9	29.9	30.0	29.9	29.8	30.0	30.0	23.9	29.9	3.01
80%	30.0	16.5	29.2	30.0	28.6	27.6	29.2	29.6	19.8	25.9	4.71
70%	30.0	13.9	27.2	30.0	26.4	24.4	25.0	28.0	16.5	20.7	5.52
60%	29.9	12.0	23.7	29.9	23.2	21.3	20.6	23.9	13.4	16.7	6.1

Table 20: EFR of Desynchronized Frame Arrivals for Statically Partitioned Buffers

The outcome of fairness listed in Table 20 shows no major changes from that of synchronized arrivals. Partitioned buffers mask the effect of desynchronized frame arrivals that was observed in case of shared bandwidth and shared buffers. Since the buffer spaces are independent, the randomized arrivals do not affect the performance. In contrast, the same scheme had shown considerable improvement in case of shared bandwidth and shared buffers, where there was contention among the sources for the shared buffer space.

The spread of frame arrivals does not show any distinct change in the deviation within a sequence aspect, as in Table 21. The arrival pattern of frames into the independent buffers does not change either the fairness or variation within a sequence, except for negligible differences.

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	1.26	0.00
90 %	0.00	0.28	0.00	0.56	0.29	0.00	0.00	4.73	3.35	0.18
80 %	0.00	1.83	1.22	2.77	1.44	0.00	0.80	6.27	4.08	1.66
70 %	0.00	3.44	1.99	4.01	1.31	0.00	2.63	6.12	4.72	3.18
60 %	0.40	4.89	1.90	5.47	1.18	0.29	3.07	5.19	4.96	4.20

Order2	Video Sequences									
Bandwidth	bla	sum	tha	suk	foo	can	jet	boe	boo	mov
100 %	0.00	1.19	0.00	0.00	1.26	0.00	0.00	0.00	0.00	0.00
90 %	0.00	4.70	0.00	0.00	3.43	0.00	0.40	0.18	0.22	0.18
80 %	0.00	6.23	0.00	0.85	4.09	1.12	2.62	1.69	1.74	1.58
70 %	0.00	6.13	0.00	2.65	4.65	2.15	3.99	3.24	3.47	1.69
60 %	0.40	5.39	0.29	3.00	5.14	1.95	5.44	4.22	4.81	1.22

Order3	Video Sequences									
Bandwidth	bla	foo	boe	tha	boo	jet	can	suk	sum	mov
100 %	0.00	1.29	0.00	0.00	0.00	0.00	0.00	0.00	1.12	0.00
90 %	0.00	3.37	0.18	0.00	0.25	0.49	0.00	0.00	4.68	0.22
80 %	0.00	3.96	1.68	0.00	1.80	2.77	1.14	0.88	6.25	1.46
70 %	0.00	4.69	3.24	0.00	3.38	4.04	1.99	2.71	6.15	1.15
60 %	0.40	5.04	4.22	0.26	4.79	5.44	1.67	3.17	5.18	1.35

Table 21: SD-of-EFR of Desynchronized Frame Arrivals for Statically Partitioned Buffers

6.4 Summary

We notice that the performance does not differ much between desynchronized arrivals and synchronized arrivals in case of statically split buffers and shared bandwidth. Below we list our observations highlighting the effect of partitioned buffers on the two objectives.

- While split buffers provide appreciable improvement in terms of *fairness* under synchronized arrivals, the large fluctuations observed in case of shared link and shared buffers under the same arrival pattern is considerably reduced. Desynchronized arrivals do not result in any notable performance change. In-spite of the improvement achieved, there is still a possibility of further enhancing the performance.

- *Variation within a sequence* is influenced by the independent buffer spaces as observed in the case of synchronized arrivals. In comparison with the desynchronized frame arrival based shared link and shared buffers experiment, the results show minor difference.

- Much of the improvement observed in case of synchronized arrivals is attributed to the partitioned buffers, as the desynchronized arrivals yield no significant change in performance.

Chapter 7

Static Bandwidth Partition

In the last Chapter, we noticed a fair amount of improvement in performance when the buffer space was split among the clients. Though buffer partition helps improve the performance, it is not sufficient. We encounter undesirable effects due to shared bandwidth which leads to interference among the streams. At the same time, the outcome implies that the performance improves under independent resource allocation. Thus, in this Chapter, we investigate whether bandwidth partitioning can lead to an improved fairness. Based on this idea, we propose to split bandwidth as each stream would be independent, eliminating any interference from others.

The current set of experiments employs an approximation of Generalized Processor Sharing scheduling technique such that both bandwidth and buffers are partitioned statically among the streams. When a new frame arrives at the gateway, the FAC checks for the availability of adequate buffers in the independent queue, if found, the frame is appended to the queue. If not, the frame is dropped. The enqueued video frames are served in a weighted fair share manner, an approximation of the Generalized Processor Sharing scheme.

7.1 Bandwidth Partitioning

We propose to split the bandwidth in two different ways. In the first scheme, we partition the available bandwidth *proportional to the average bandwidth of each sequence* (Scheme 1). In the second scheme, we partition the available bandwidth based on the *apriori determined target* EFR (Scheme 2).

Bandwidth allocation based on Scheme 1 guarantees that every sequence gets a weighted share under the prevailing resource crunch and is simple to implement. On the other hand, the second possibility involves determining a common EFR for all sequences, for which the sum of corresponding bandwidths is equivalent to available bandwidth.

In Chapter 4, when each sequence was run independently over the entire range of their respective average bandwidth, we noticed that the EFR achieved decreased almost linearly with decrease in bandwidth. Such a behaviour serves as the motivating factor for Scheme 2. The output EFR values of this experiment are employed as the preprocessed data which in fact forms the input matrix. Using the matrix, the algorithm interpolates to find the split for a sequence. This is a two step process: in the first step, a region is found among sample points where the currently tried target EFR is located. In the next step, corresponding required bandwidth is found with few simple mathematical operations. The target partition is obtained when the sum of individual splits does not exceed the available bandwidth.

This scheme provides a better bandwidth split for a particular sequence compared to Scheme 1 by directly addressing the desired goal, but it is expensive as more information is required. We present the complete process for Scheme 2 in Algorithm 1. We list sample bandwidth partitions obtained from both the schemes in Tables 22 and 23. A notable feature of these bandwidth partition schemes is the variation in allocation of link capacity to each sequence. For instance, the sequence `sum` is alloted 0.984528 Mbps in Scheme 1 whereas it is 0.961303 Mbps in Scheme2.

7.2 Overview of Experiments

Experiments in the previous two sets are split into two categories: synchronized arrivals and desynchronized frame arrivals. In the current set of experiments, as both bandwidth and buffer space are independent for each sequence, we do not observe any substantial change in performance between the synchronized and desynchronized frame arrivals. Thus, we choose the desynchronized frame arrival case to reflect a realistic environment. Furthermore, due to separate buffer and bandwidth space, the positional effect of a sequence in an order becomes insignificant. Hence, instead of three orders, we have chosen only one order in the form of Order1.

Algorithm 1: Static Bandwidth Partition Based on Predetermined EFR.

Inputs. The Available Bandwidth for n sequences.
For each sequence:
The array BW[ ] containing sample (SP) bandwidth values.
The array EFR[ ] containing EFR achieved for BW[ ] elements.
Outputs. The partitioned bandwidths in the array PartBW[ ].
begin algorithm
Target EFR = 30.0; Total bandwidth = 0; Terminate = FALSE; Step size = 0.1;
while (not Terminate)
for (seq = 0 to n-1) **do**
found = FALSE
for (points = 0 to (SP - 1 **or** not found)) **do**
if (EFR[points] < Target EFR **and** not found) **do**
position = points
found = TRUE
endif
endfor
min EFR = EFR[position]
max EFR = EFR[position - 1]
min bandwidth = BW[position]
max bandwidth = BW[position - 1]
bandwidth interval = max bandwidth - min bandwidth
EFR interval = max EFR - min EFR
EFR difference = Target EFR - min EFR
BW difference = (bandwidth interval / EFR interval) × EFR difference
Interpolated bandwidth = min bandwidth + BW difference
Total bandwidth = Total bandwidth + Interpolated bandwidth
PartBW[seq] = Total bandwidth
endfor
if (Total bandwidth > Available Bandwidth)
then Target EFR = Target EFR - Step size
else Terminate = TRUE
end while
{ split any residual bandwidth among the sequences }
if (sum of PartBW < Available Bandwidth) **do**
for (seq = 0 to n-1) **do**
PartBw[seq] = (PartBw[seq] / sum of PartBW) × Available Bandwidth
endfor
endif
return PartBW[ ]
end algorithm

Video Name	Bandwidth Split for Scheme 1	Bandwidth Split for Scheme 2
bla	0.309096	0.315775
boo	0.503712	0.496357
can	0.560952	0.562591
jet	0.538056	0.539628
mov	0.675432	0.690759
tha	0.274752	0.272387
suk	0.492264	0.484854
sum	0.984528	0.961303
foo	1.064664	1.078399
boe	0.480816	0.482220

Video Name	Bandwidth Split for Scheme 1	Bandwidth Split for Scheme 2
bla	0.274752	0.284323
boo	0.447744	0.438264
can	0.498624	0.507260
jet	0.478272	0.473949
mov	0.600384	0.621299
tha	0.244224	0.242235
suk	0.437568	0.437805
sum	0.875136	0.840303
foo	0.946368	0.957402
boe	0.427392	0.427624

Table 22: Static Bandwidth Split for 5.884272 Mbps (left) and 5.230464 Mbps (right)

Video Name	Bandwidth Split for Scheme 1	Bandwidth Split for Scheme 2
bla	0.240408	0.253025
boo	0.391776	0.385875
can	0.436296	0.449100
jet	0.418488	0.405601
mov	0.525336	0.552905
tha	0.213696	0.211338
suk	0.382872	0.390059
sum	0.765744	0.713824
foo	0.828072	0.837078
boe	0.373968	0.377850

Video Name	Bandwidth Split for Scheme 1	Bandwidth Split for Scheme 2
bla	0.206064	0.221021
boo	0.335808	0.335809
can	0.373968	0.391832
jet	0.358704	0.336563
mov	0.450288	0.482972
tha	0.183168	0.182116
suk	0.328176	0.340442
sum	0.656352	0.594601
foo	0.709776	0.709779
boe	0.320544	0.327712

Table 23: Static Bandwidth Split for 4.576656 Mbps (left) and 3.922848 Mbps (right)

Order1	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
Buffer Size	1476	1035	406	2695	915	1565	1080	3732	4652	2886

Table 24: Configuration Parameter for Statically Partitioned Bandwidth

We maintain independent buffers for each source as in the case of last set of experiments. The independent buffer sizes for all ten streams are reproduced from Chapter 4 in Table 24. The bandwidth is partitioned according to the two schemes. Such a partition is realized in practice using the Weighted Fair Queueing, while in our simulations we separately run each sequence for a particular bandwidth capacity and buffer value. Thus, we have ten runs for a particular bandwidth value. Both these forms approximate the Generalized Processor Scheduling technique. We employ the partitioned bandwidth values presented in Tables 22 and 23 for the current experiment.

7.3 Results

We present results for the two bandwidth partition schemes with independent buffers under desynchronized frame arrivals in this section.

7.3.1 Frame Arrival Desynchronization

We conducted experiments with the partitioned bandwidths for schemes 1 and 2. The fairness values are depicted in Table 25.

The results of both these mechanisms show remarkable improvements when compared to the previous two sets of experiments. In the first two sets, we observed large differences in fairness among the ten sequences. For instance, in the first set we observed a difference of around 10 fps between the highest and lowest EFR recorded for 80% bandwidth; in the second set, wherein buffer space was partitioned we observed a similar behaviour with minor fluctuations. In this set, as each sequence is completely independent we observe a high EFR difference of around 3 fps in case of 60% bandwidth for scheme 1. In terms of variation in fairness, we notice that there is a substantial reduction. The FI is less than 1.0 for all the bandwidth points in the constrained region.

Order1	Video Sequences										FI
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	0.03
90%	26.8	27.7	27.3	27.3	26.8	27.6	27.8	27.9	27.1	27.4	0.39
80%	23.9	25.2	24.3	24.9	23.9	24.8	24.7	25.5	24.5	24.7	0.51
70%	20.9	22.3	21.4	22.5	20.9	22.2	21.6	23.1	21.8	21.8	0.7
60%	17.9	19.2	18.3	20.1	17.9	19.3	18.5	20.7	19.2	18.8	0.91

Order1	Video Sequences										FI
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	30.0	30.0	29.1	30.0	30.0	30.0	30.0	29.8	28.5	0.53
90%	27.4	27.4	27.4	27.4	27.5	27.4	27.4	27.4	27.4	27.4	0.03
80%	24.7	24.7	24.8	24.7	24.7	24.7	24.7	24.7	24.7	24.7	0.03
70%	22.0	22.0	21.9	22.0	22.0	22.0	22.0	22.0	22.1	22.0	0.05
60%	19.2	19.2	19.2	19.3	19.2	19.2	19.2	19.2	19.2	19.2	0.03

Table 25: EFR of Desynchronized Frame Arrivals for Static Bandwidth Partition Mechanisms: Scheme 1 (top) and Scheme 2 (bottom)

Scheme 2 achieves the best results in comparison with the three sets discussed so far. The results show near identical EFRs for all sequences and for all bandwidth values in the constrained region. The difference is a negligible 0.1 fps in the 90% to 60% constrained bandwidth region. This results in a negligible FI.

On variation within a sequence aspect, the results in Table 26 show negligible difference between the two partition schemes. With regard to the previous two sets we observe only minor differences. In effect, the variation within a sequence still shows large variations. This behaviour is not unexpected, as bandwidth is fixed for the entire duration. Such an allocation would result in all frames being sent in some intervals, while in some others it may not. We thus note that, there is still a possibility to further improve the deviations within a stream.

7.4 Summary

We have presented the partitioned bandwidth as well as the partitioned buffers case reflecting the advantage of Scheme 2 over Scheme 1. Such an outcome leads to significant gains in terms of fairness, with large variations in deviation within a sequence. The fairness achieved is the best of the three sets. On the other hand, the deviation

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	0.31	0.00	0.00	0.00	0.00	0.00	0.00
90 %	2.43	2.81	1.67	2.73	0.56	2.84	3.02	3.14	2.64	3.58
80 %	2.37	4.23	1.77	4.24	0.60	3.77	2.99	4.78	3.89	4.50
70 %	2.08	5.03	1.72	5.28	0.62	4.83	2.60	6.02	4.67	4.69
60 %	1.86	4.50	1.49	6.37	0.50	5.25	2.32	6.37	5.23	4.42

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.00	0.00	1.22	0.00	0.00	0.00	0.00	0.50	2.79
90 %	2.32	3.00	1.63	2.66	0.65	2.91	3.18	3.61	2.50	3.52
80 %	2.46	4.38	1.73	4.27	0.63	3.81	2.98	5.18	3.86	4.52
70 %	2.13	5.05	1.70	5.55	0.43	4.90	2.67	6.27	4.53	4.73
60 %	2.00	4.50	1.64	6.64	0.56	5.23	2.34	6.53	5.23	4.49

Table 26: SD-of-EFR of Desynchronized Frame Arrivals for Static Bandwidth Partition Mechanisms: Scheme 1 (top) and Scheme 2 (bottom)

within a sequence results indicates further need for improvement. We summarize the observations in the following:

- In terms of *fairness*, the outcome is remarkable and surpasses the previous two sets.
- In terms of *Variation within a sequence*, we observe large variations which differ from the previous set with minor difference. Thus, there is potential for improvement in this aspect.
- The current set of experiments is characterized by complete elimination of interference among the sequences due to shared link or shared buffer space thereby resulting in a significant improvement in fairness.

Chapter 8

Dynamic Bandwidth Partition

In the previous Chapter, static partitioning of the bandwidth produced significant gains in fairness. At the same time, large deviations within a sequence still persisted. We obtained such an undesirable outcome because of the variations in frames sent due to static bandwidth allocation. On the contrary, we noticed that reduced interference produces improved performance in connection with fairness. In this Chapter, we examine whether time varying partitioning of bandwidth can improve the deviations in a sequence.

In the current set of experiments, we employ dynamically partitioned bandwidth such that the bandwidth is shared among the competing streams based on their need over specific time intervals. In our case, such an interval is fixed at one second. On the contrary, buffers are statically partitioned. The gateway accepts a new frame if there is sufficient space in the independent queue, otherwise, the frame is dropped. The enqueued frames are forwarded onto the link based on the bandwidth, which changes for every second.

8.1 Dynamic Bandwidth Partitioning Scheme

We present a scheme wherein, the bandwidth is dynamically partitioned over one second quanta. In effecting such a partition, we take into account the prevailing requirements of the sequence to achieve the target quality. We obtain partitioned bandwidth for every second, when the sum of bandwidth requirement for all sequences is less than the available bandwidth.

In the previous set, we observed high deviations as frames dropped varied over time owing to fixed bandwidth. We address such an effect by distributing the bandwidth for each one second quantum by taking into account the prevailing needs of each sequence, to attain a common target EFR for all streams. In effecting partitioning this way, we exercise greater control over the manner bandwidth is allocated. The dynamic partition algorithm is similar to Scheme 2-algorithm presented in the previous Chapter. The only difference is that partitioning is effected every second rather than once for the entire duration of transmission. We present the required steps in Algorithm 2.

We derive preprocessed data for the algorithm, based on the behaviour of near linear drop in EFR with decrease in bandwidth, mentioned in the previous Chapter. The data are obtained in terms of the EFR achieved per second for sample bandwidth points spread over the entire average bandwidth range of each sequence. These data, in fact, form the region within which the algorithm interpolates to find the split for all sequences during the one second quanta. Such a split is governed by the target EFR under consideration, which ranges from a peak of 30 fps and lower. The process of determining best bandwidth match for the target EFR is identical to that of Algorithm 1. The target partition for each second is obtained when the sum of individual splits does not exceed the available bandwidth.

The dynamic bandwidth partition scheme provides a better bandwidth split when compared to static split schemes by directly considering the desired goal. But, it is expensive as it needs more information (per second EFR values for a bandwidth point) than the second scheme of static bandwidth partition (single EFR value for a bandwidth point).

In order to illustrate the operation of the dynamic bandwidth partition algorithm, we present bandwidth split for 80% of *Average Bandwidth* case in graphs 28 through 32.

The plots depict bandwidth variations of sequences over one second quanta, which varies among the highest and the lowest bandwidth requirement. For instance, in Table 28, the sequence boo has a maximum bandwidth of approximately 0.55 Mbps on few occasions, while it shows different needs in course of the transmission time – requiring well below 0.35 Mbps in some instances. For the same sequence, static bandwidth allocation policy would have allocated 0.494 Mbps under Scheme 2 for the entire duration of the transmission and this illustrates the main advantage of dynamic

Algorithm 2: Predetermined EFR Based Dynamic Bandwidth Partition.

Inputs. The Available Bandwidth for n sequences.
For each sequence:
The array BW[ ], with sample (SP) bandwidth values.
The array EFR[ ], with each row containing EFR achieved for BW[ ] elements over one second quanta.
Outputs. The array DBW[ ] with dynamically split bandwidths for one second quanta.

```
begin algorithm
  for (time = 0 to total time) do
  Target EFR = 30.0; Terminate = FALSE; Step size = 0.1;
    while (not Terminate)
      for (seq = 0 to n-1) do
        found = FALSE;  Total bandwidth = 0
        for (points = 0 to (SP - 1 or not found)) do
          if (EFR[points] < Target EFR and not found) do
            position = points {points modified to access respective time and EFR}
            found = TRUE
          endif
        endfor
        min EFR = EFR[position];  max EFR = EFR[position - 1]
        min bandwidth = BW[position];  max bandwidth = BW[position - 1]
        bandwidth interval = max bandwidth - min bandwidth
        EFR interval = max EFR - min EFR
        EFR difference = Target EFR - min EFR
        BW difference = (bandwidth interval / EFR interval) × EFR difference
        Interpolated bandwidth = min bandwidth + BW difference
        Total bandwidth = Total bandwidth + final bandwidth
        DBW[seq] = Total bandwidth
      endfor
      if (Total bandwidth > Available Bandwidth)
      then Target EFR = Target EFR - Step size
      else Terminate = TRUE
    end while
    { distributes any residual bandwidth among all sequences }
    if (sum of DBW[ ] < Available Bandwidth) do
      for (seq = 0 to n-1) do
        DBW[seq] = (DBW[seq] / sum of DBW[ ]) × Available Bandwidth
      endfor
    endif
    return DBW[ ]
  end for
end algorithm
```

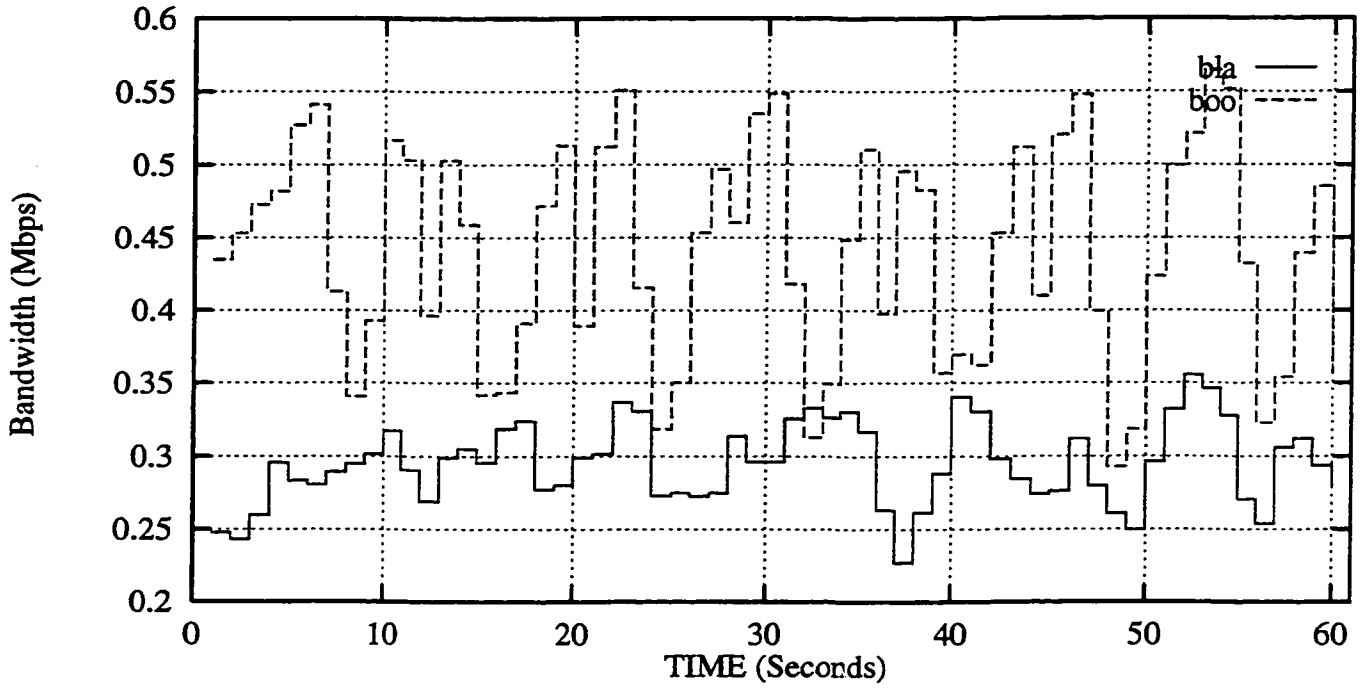


Figure 28: Dynamically Partitioned Bandwidth for bla and boo Sequences

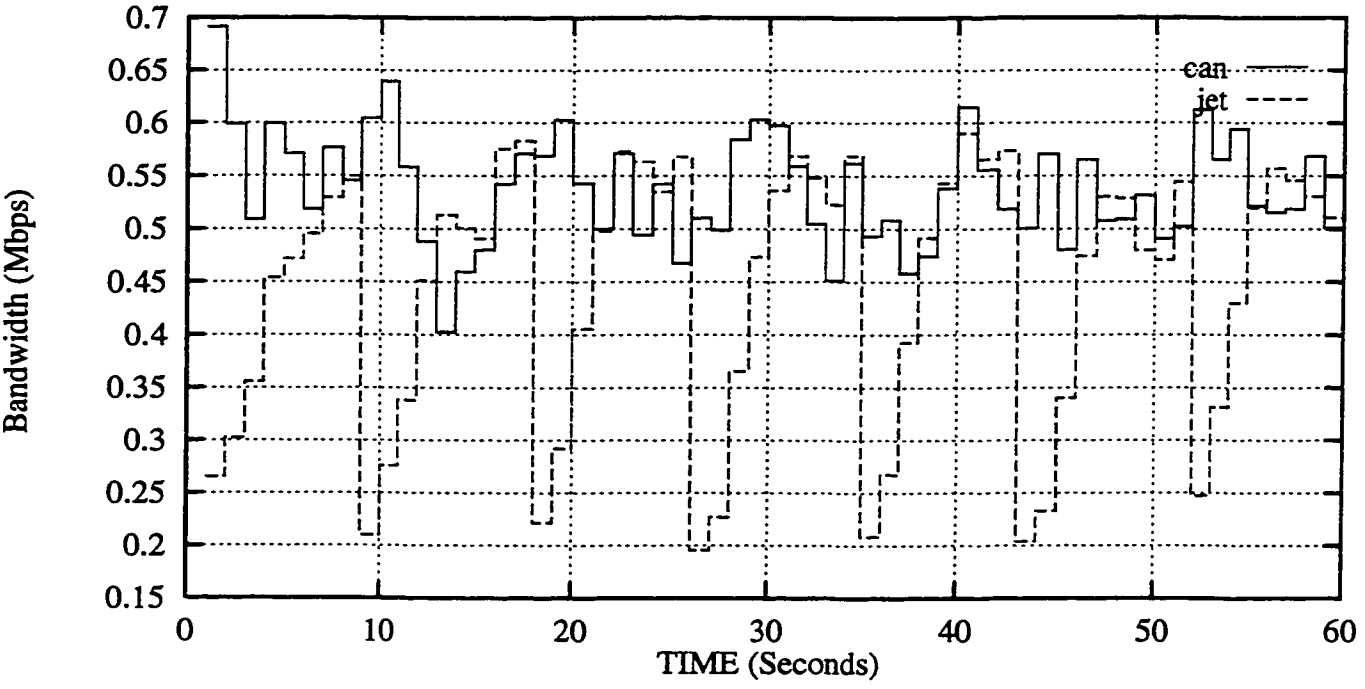


Figure 29: Dynamically Partitioned Bandwidth for can and jet Sequences

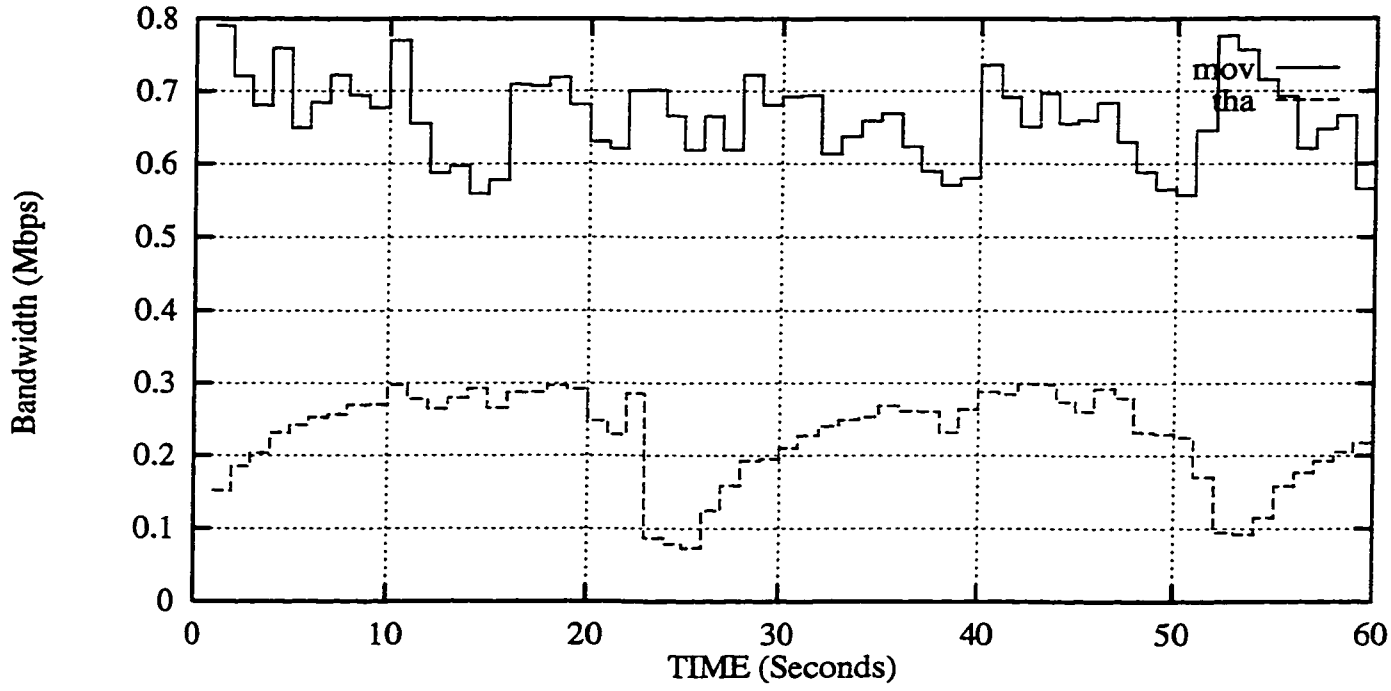


Figure 30: Dynamically Partitioned Bandwidth for mov and tha Sequences

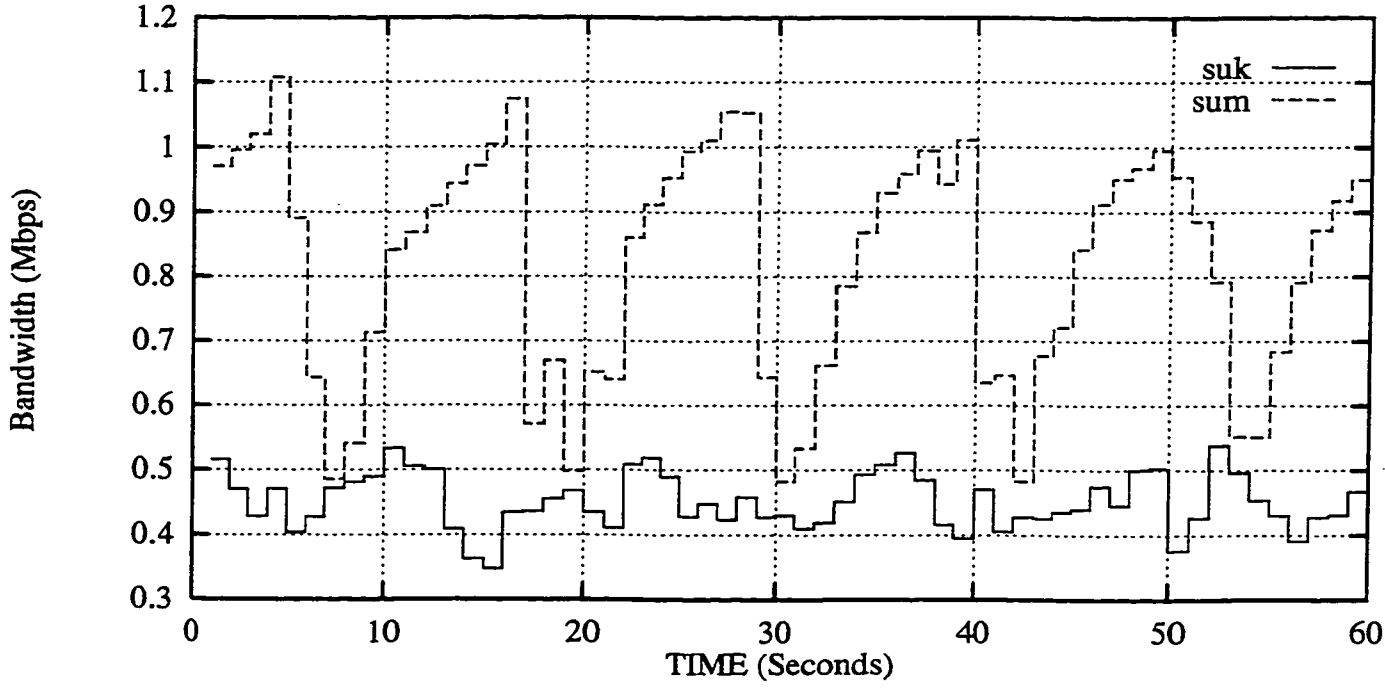


Figure 31: Dynamically Partitioned Bandwidth for suk and sum Sequences

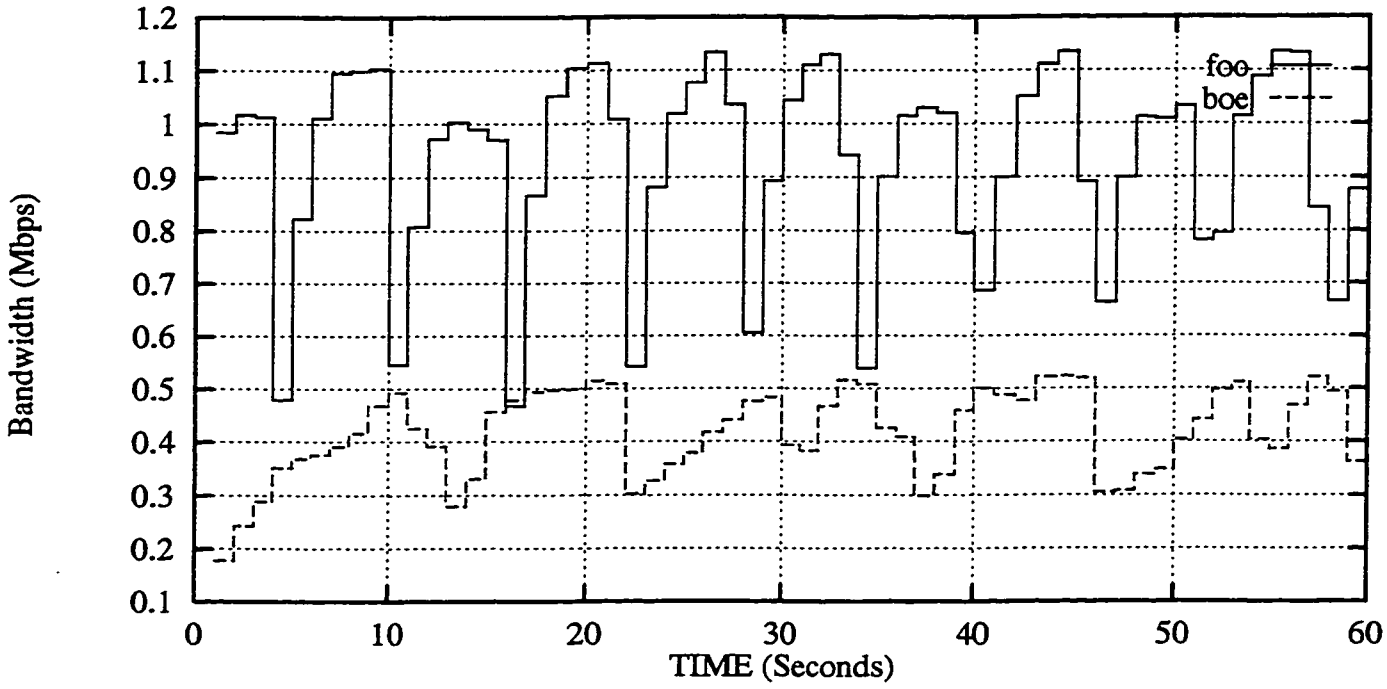


Figure 32: Dynamically Partitioned Bandwidth for `foo` and `boe` Sequences

bandwidth partition algorithm. The plots for other constrained bandwidth points are included in the appendix.

8.2 Overview of Experiments

As in the previous set of experiments, we do not notice any major change in performance between synchronized arrivals and desynchronized frame arrivals. Thus, we opt for desynchronized frame arrivals to reflect a realistic environment. Also, the position of a stream in an order does not matter as in the previous set and hence we choose Order1 for our experiments.

The buffers are statically partitioned and we employ the same independent buffer sizes that were used in the previous set. The bandwidth, on the other hand, is partitioned dynamically. In practice, this can be realized using Weighted Fair Queueing, but instead of a single bandwidth value for the entire transmission, the scheduler has to change link capacity every second. In our simulations, we run each sequence separately. The dynamic bandwidth values are supplied as input to the simulator.

Order1	Video Sequences										FI
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe	
100%	30.0	29.8	30.0	28.6	30.0	28.6	30.0	27.8	29.5	29.7	0.78
90%	28.6	27.5	29.2	26.2	29.8	26.0	28.5	25.8	27.2	27.3	1.38
80%	25.6	24.4	25.9	23.6	26.0	23.6	25.6	24.2	24.3	24.4	0.93
70%	22.1	21.5	22.2	21.1	22.2	21.2	22.2	21.6	21.2	21.7	0.45
60%	18.8	18.7	18.9	18.4	18.8	18.5	18.8	18.6	18.2	18.7	0.22

Order1	Video Sequences									
Bandwidth	bla	boo	can	jet	mov	tha	suk	sum	foo	boe
100 %	0.00	0.56	0.00	2.47	0.00	2.25	0.00	2.31	0.91	0.87
90 %	1.82	2.86	1.21	3.70	0.57	3.31	2.39	3.38	2.18	3.10
80 %	1.76	2.81	2.08	4.22	2.02	3.44	2.39	3.60	2.88	2.61
70 %	1.75	2.03	1.91	3.89	1.89	2.91	1.87	2.63	2.69	1.94
60 %	1.63	1.64	1.68	3.16	1.74	2.08	1.66	1.77	2.65	1.55

Table 27: EFR (top) and SD-of-EFR (bottom) of Desynchronized Frame Arrivals for Dynamically Split Bandwidth

8.3 Results

The experiments in this section are characterized by the dynamically partitioned bandwidth values when compared to previous set of experiments.

8.3.1 Frame Arrival Desynchronization

We deliberately employ the same constrained bandwidth region as in the previous sets to enable us to compare the performance. We present the fairness results in Table 27. The EFRs are in the expected range with minor differences. The fairness obtained is comparable to that of the third set wherein both bandwidth and buffers were statically split. The FI is less than 1.0 except 90% bandwidth case, which is similar to that of Scheme 1 of the statically partitioned bandwidth. Prominent streams which exhibit variations in fairness include `jet`, `tha`, `sum` and `foo`. Such a behaviour is attributed to the frame size characteristics, which consistently deviate away from their average frame sizes for long durations.

The outcome of deviation within a sequence shows significant improvements as can be seen in Table 27. In the previous set, we encountered high variation within

a sequence. Such a behaviour included large disparity among sequences and also a high individual value. For instance, in the last set for Scheme 2, sum had a high SD-of-EFR of 6.53 whereas it is 3.6 in the current set. Among other sequences, we observe similar improvement in case of jet.

Another key improvement is seen in terms of deviation of a particular sequence over constrained region. For example, sum had a difference of 2.92 between the lowest and highest SD in the second scheme of the previous set, while it is 1.83 in this experiment. We observe similar trend in other sequences.

8.4 Summary

We observed significant gains in fairness when bandwidth was split in the third set. Such a gain in fairness was accompanied with a high deviation in each sequence. The current set of experiments address this issue yielding major improvements, with minor variations in fairness. Thus, dynamic partitioning answers the two objectives we set out in the problem statement. Below we capture the significance of dynamic bandwidth partition algorithm.

- The *fairness* achieved is comparable with that of statically partitioned bandwidth case, but with minor difference.
- The *deviation within a sequence* improves significantly, answering the main thrust of current set of experiments. With dynamic bandwidth allocation, the losses are further localized to one-second intervals, thus aiding the deviation aspect.
- Results of the current set meet the objectives we outlined in the problem statement, by maximizing the minimum fairness and achieving low deviation within a sequence.

Chapter 9

Conclusions

Distributed multimedia applications are fast occupying the bandwidth on networks, which were designed to carry ordinary data. Until recently, computer networks and protocols have evolved to ensure reliable data delivery, but make virtually no assurance to an application in terms of expected throughput and delay. On the contrary, multimedia applications impose stringent demands in terms of bandwidth, delay and synchronization, including reliability.

In order to meet the specific needs of multimedia applications, recent developments in networking suggest resource reservation. Such a reservation can be provided primarily for bandwidth, but also, perhaps for buffers. Networks offer different services based on these reservations. Examples of such services include, CBR, VBR, ABR, etc., or Controlled Load.

The different media constituting multimedia applications have different performance requirements. Human visual system tests have established that video, with a tolerance level of 40 ms, is more tolerant to loss than audio. A video stream typically has time varying bandwidth requirements. Therefore, it is attractive to use a low-level network service, such as CBR to carry multiple video streams at the same time. Statistical multiplexing of multiple streams would result in lower cumulative bandwidth and buffering requirements to deliver the same quality.

We notice that much work has been done on video transmission and multiplexing of multiple streams. However, most such work has focused on near zero loss transmission.

In this thesis, we focus on the case where the bandwidth is limited. That is, consider the case when N streams are transmitted over a link of bandwidth B, such

that the cumulative bandwidth requirements (average) of all streams is higher than B. In this case, losses are inevitable. In such an environment, the challenge is to design queueing policies to achieve maximal, but fair, performance for all video streams. As video data is subjective in nature, we employ Effective Frame Rate (EFR) as a performance measure. The EFR varies over time. Therefore, the objective is to ensure that at any time the EFR of all streams is similar and that there is no bandwidth lost.

We conducted four sets of experiments with different policies of managing the queue and bandwidth. The aim was to see what kind of queueing and bandwidth management policies are needed to achieve the desired objectives. In the first set, we use FIFO queueing policy for shared buffers with shared bandwidth. In the second set, we employ FIFO queueing policy for partitioned buffers along with shared bandwidth. In the third set, we use WFQ policy by way of partitioned buffers, and in addition, split bandwidth. In the last set, we retain the partitioned buffers and the bandwidth was split dynamically.

Our results indicated that, explicit bandwidth management is required to ensure good and fair performance for all streams. That is, simple FIFO scheduling is not enough to ensure fair quality to the competing streams even when the buffers are partitioned. Overall, the best performance was achieved when bandwidth was allocated to the different streams to achieve a maximal quality level for all streams, and when this allocation was dynamically tuned to match the time varying characteristics of the video sequences.

9.1 Limitations and Directions for Future Work

In our work, we have assumed many things: (1) stored video, (2) all video sequences had the same nominal frame rate (30fps), (3) all video frames were independent frames (no temporal dependency), (4) complete knowledge of video characteristics known in advance, and (5) overheads are not accounted for. Also, the drop entity in our case – frame – is too coarse a granularity for drops. Instead, low level video components such as, slice and macroblock level drops, utilizing Forward Error Correction techniques would be more beneficial. Nevertheless, we expect that the key result of this thesis, i.e., bandwidth must be explicitly allocated (in a time varying manner) to achieve

a maximal quality, should still hold. Clearly, in reality, one must account for all of these factors.

One way to view it is to assume a time varying function $Q\_t(B)$ that gives the achievable quality for a video stream when bandwidth B is allocated to it. This function should take into account the nominal frame rate, the temporal dependencies, etc., and may be approximated. This function can then be used to drive the bandwidth allocation online, in a similar manner as we have done.

A major issue which can be pursued is the development of practical algorithms utilizing the idea of dynamic bandwidth allocation. Clearly, for this to work, we need the quality functions (as above) that reflect real, perceived quality as closely as possible, and not assuming the simplifying things that we used. In another item for future work, as the nature of quality functions will affect the dynamic bandwidth allocation algorithm, the complexity of such algorithms will become an issue. Finally, it would be desirable to see how this can be applied to online video streams – i.e., how can the quality functions be approximated for such streams (perhaps, based on recent past).

Appendix A

Dynamic Bandwidth Partition Plots

As mentioned in Chapter 8, in this appendix, we include plots for four sample points in the constrained bandwidth region, which are split according to the dynamic bandwidth partition algorithm (Algorithm 2). In these plots, we notice similar split pattern across different sample bandwidth points for a particular sequence.

A.1 Case: 100% *Average Bandwidth*

In this section, we present the dynamically split bandwidth graphs for 100% of *Average Bandwidth* in Figures 33 through 37.

A.2 Case: 90% *Average Bandwidth*

In this section, we include the dynamically split bandwidth graphs for 90% of *Average Bandwidth* in Figures 38 through 42.

A.3 Case: 70% *Average Bandwidth*

In this section, we present the dynamically split bandwidth graphs for 70% of *Average Bandwidth* in Figures 43 through 47.

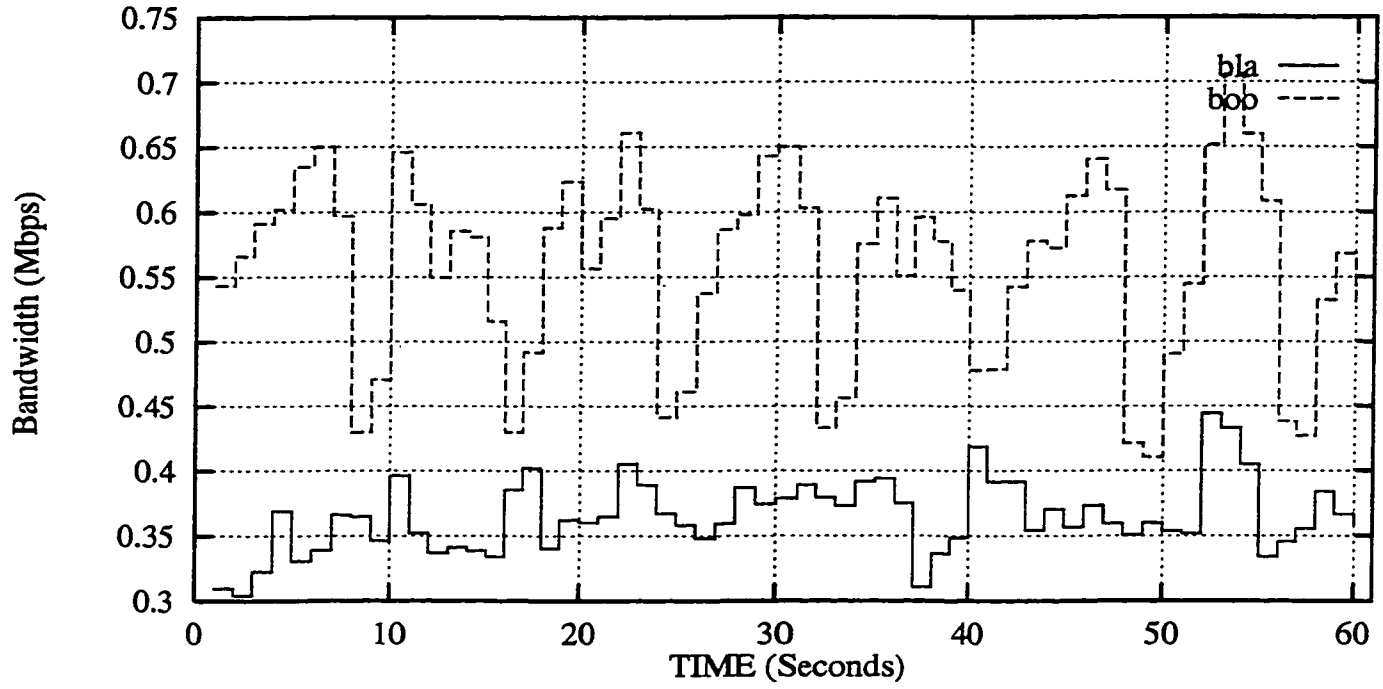


Figure 33: Dynamically Partitioned Bandwidth for `bla` and `boo` Sequences

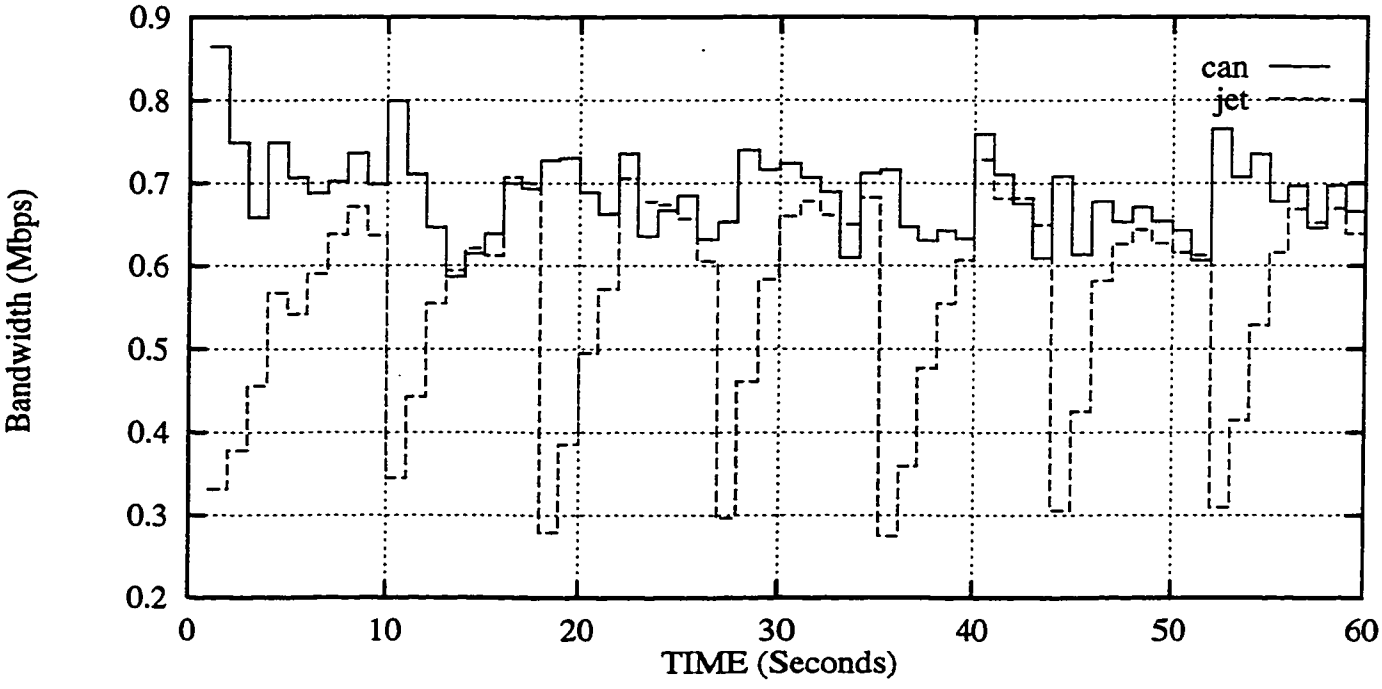


Figure 34: Dynamically Partitioned Bandwidth for `can` and `jet` Sequences

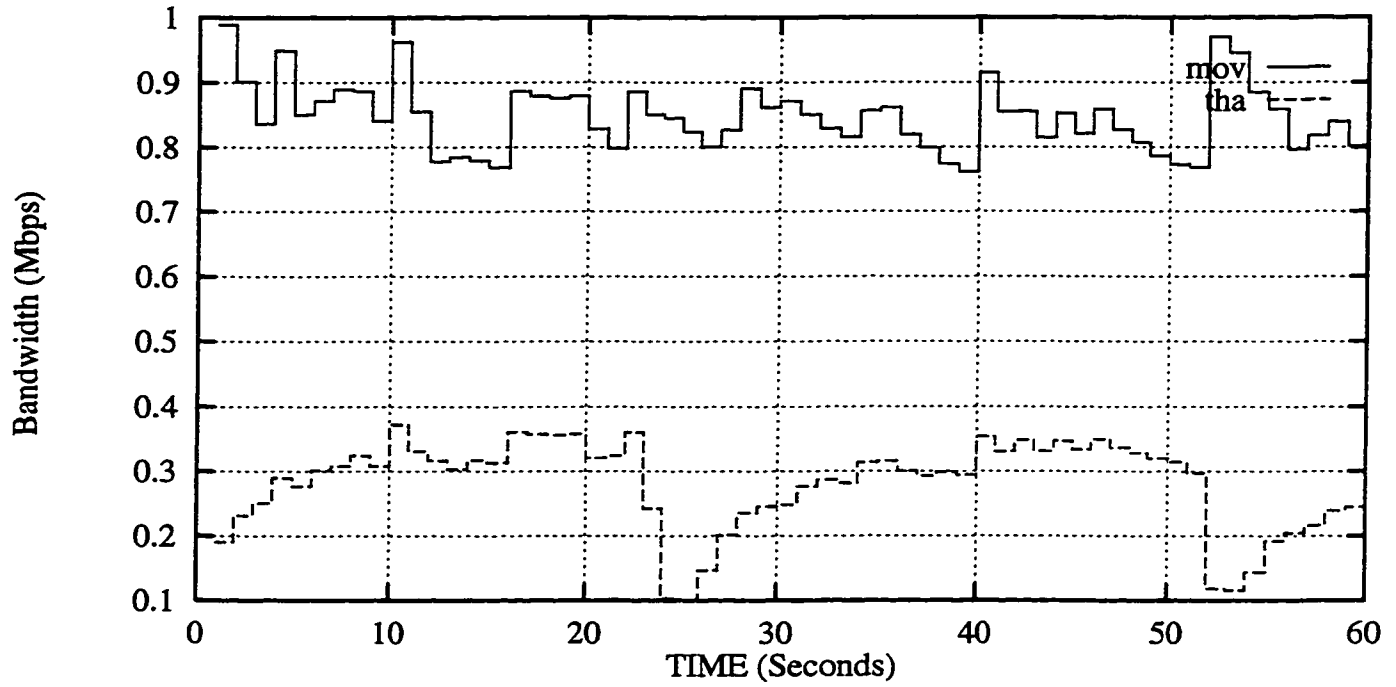


Figure 35: Dynamically Partitioned Bandwidth for mov and tha Sequences

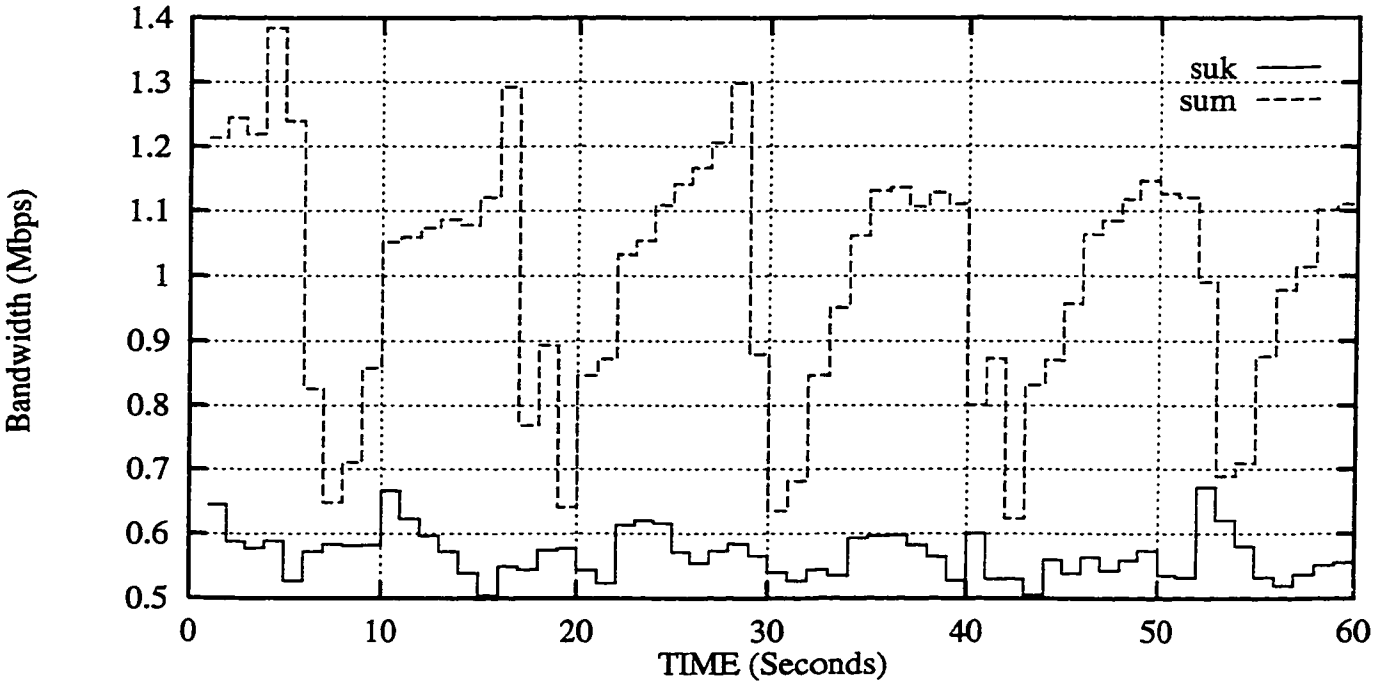


Figure 36: Dynamically Partitioned Bandwidth for suk and sum Sequences

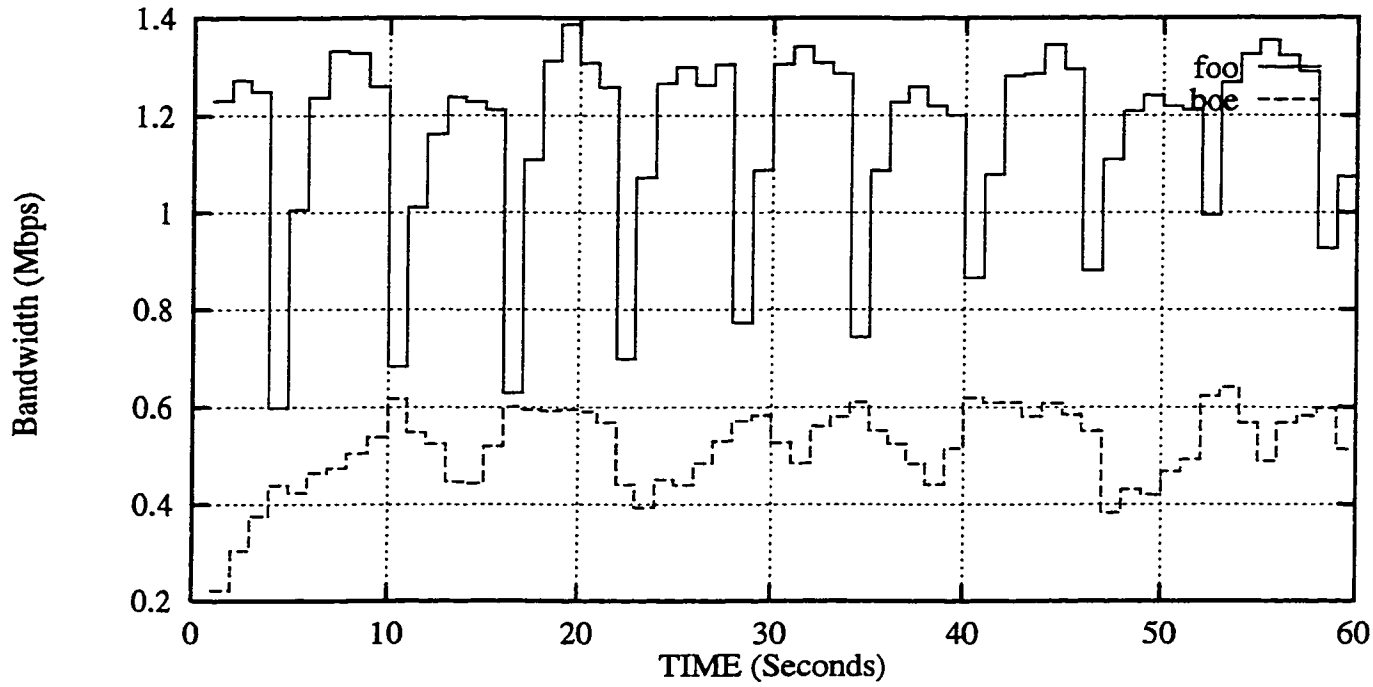


Figure 37: Dynamically Partitioned Bandwidth for `foo` and `boe` Sequences

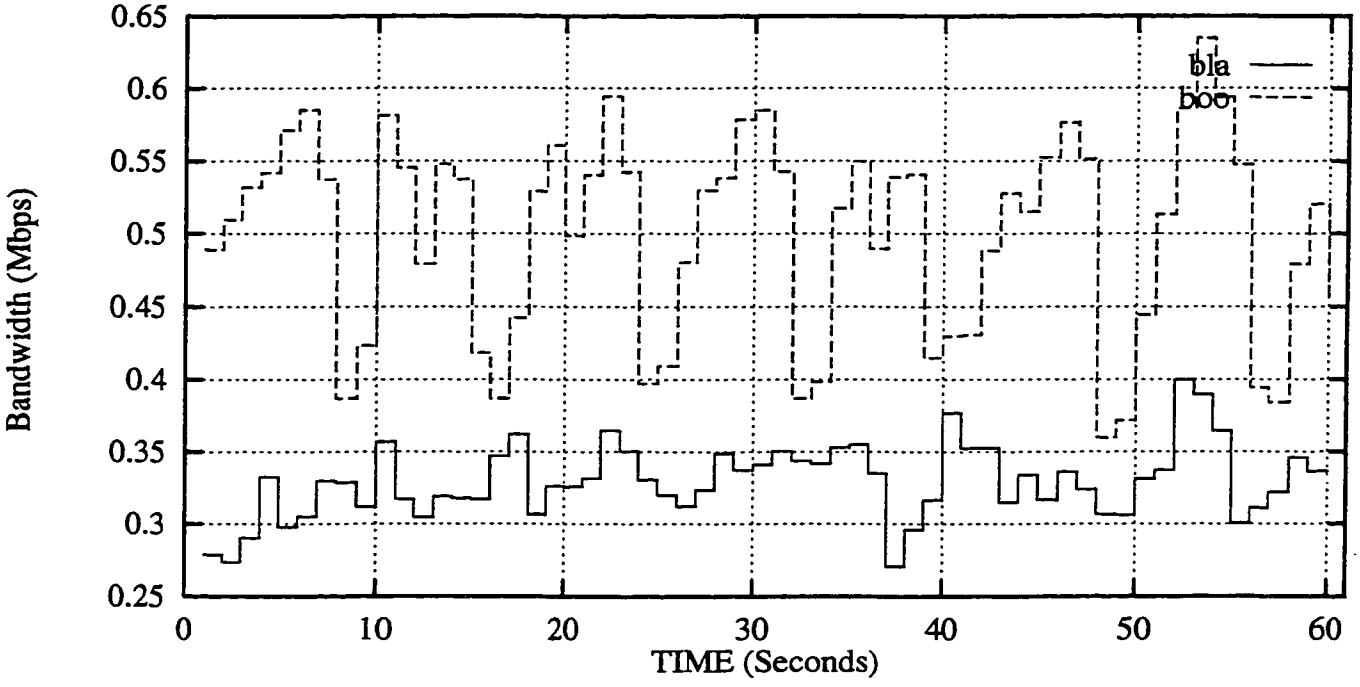


Figure 38: Dynamically Partitioned Bandwidth for `bla` and `boo` Sequences

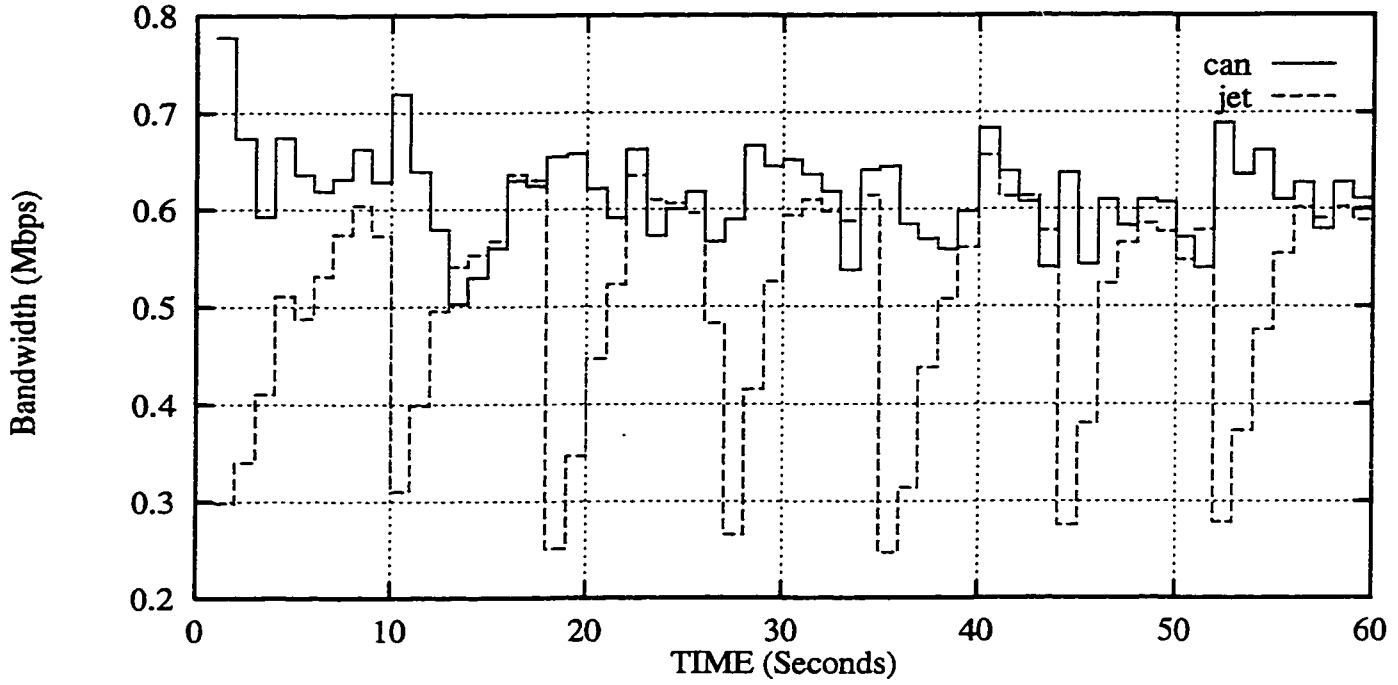


Figure 39: Dynamically Partitioned Bandwidth for `can` and `jet` Sequences

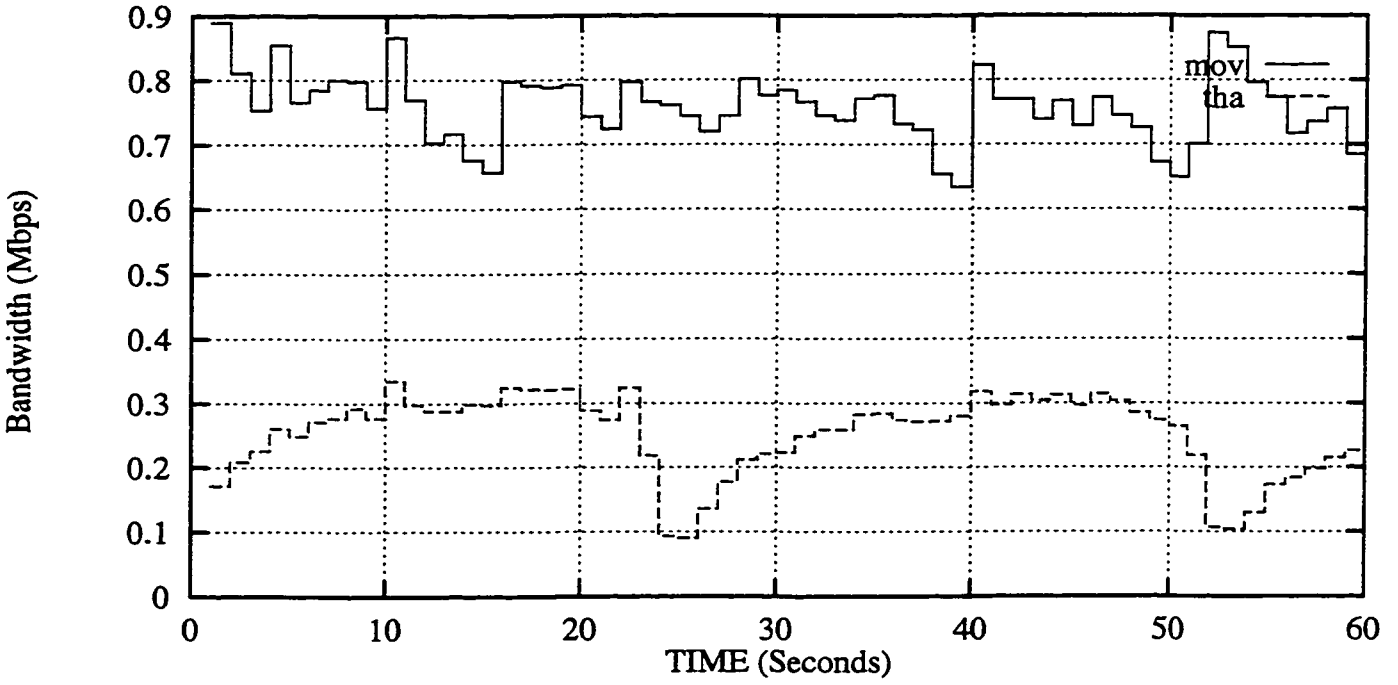


Figure 40: Dynamically Partitioned Bandwidth for `mov` and `tha` Sequences

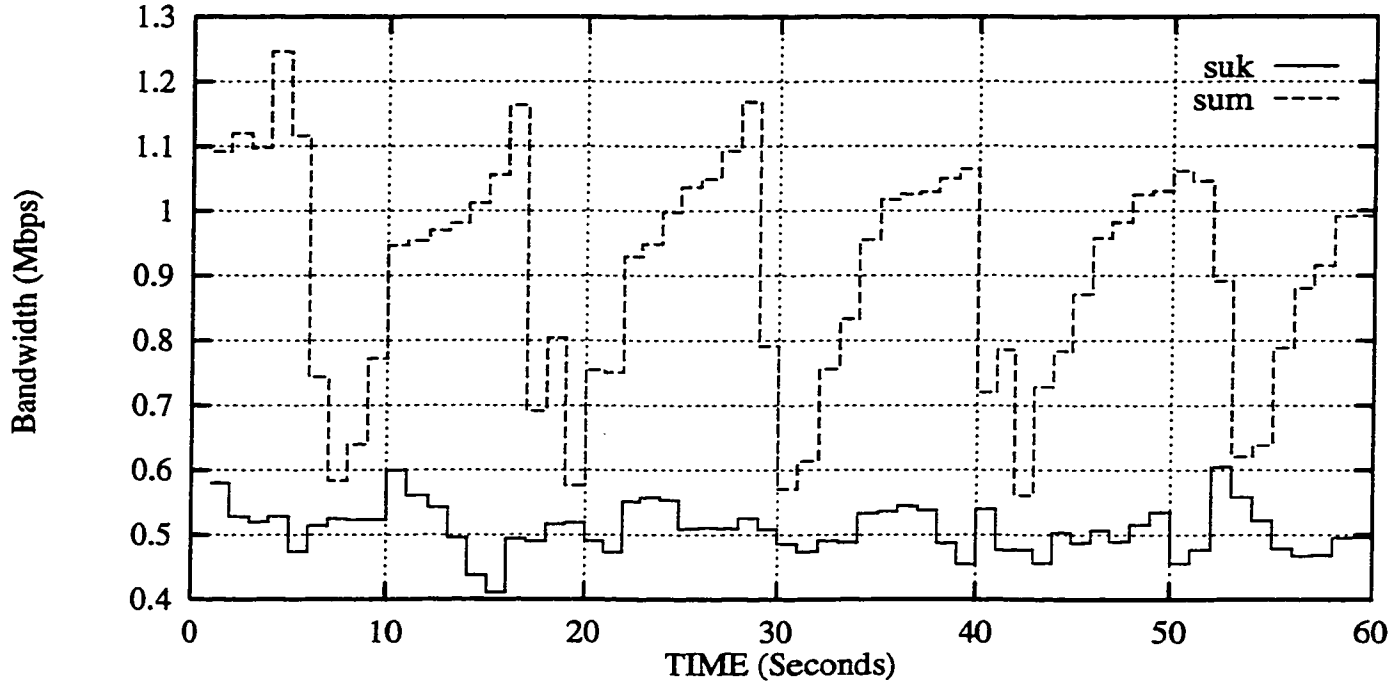


Figure 41: Dynamically Partitioned Bandwidth for suk and sum Sequences

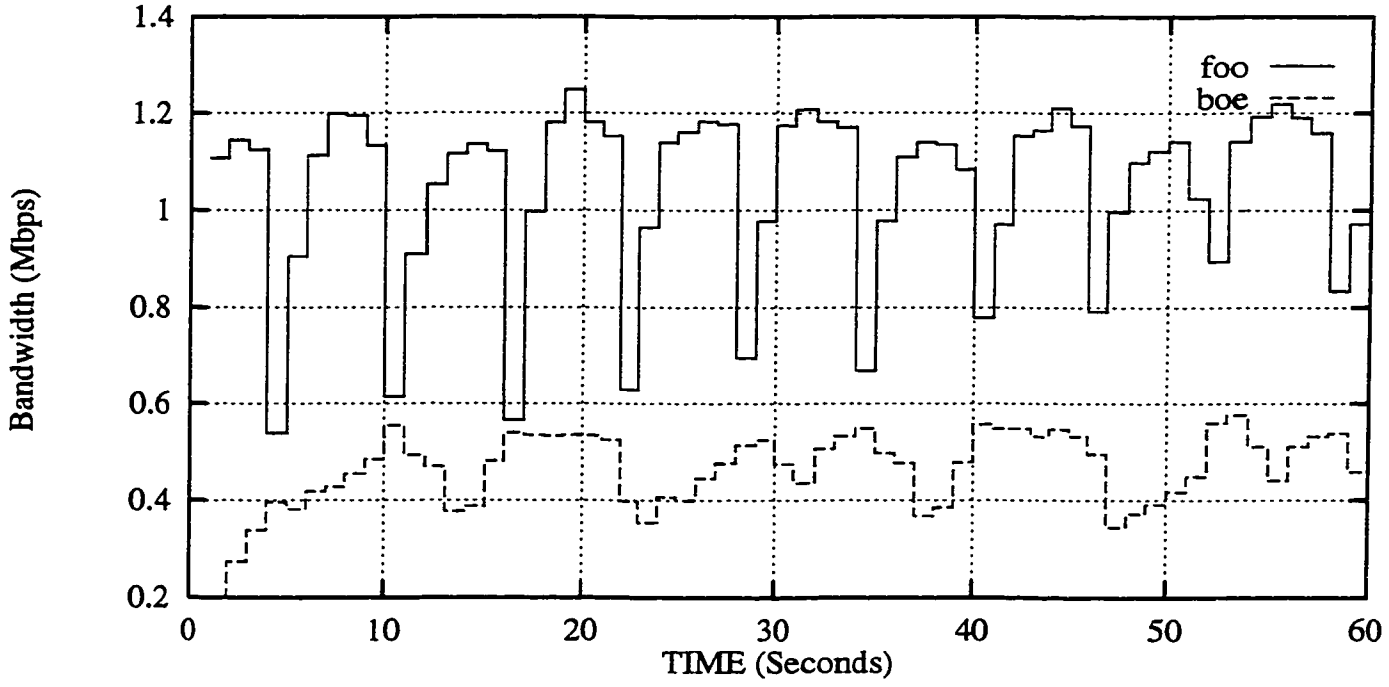


Figure 42: Dynamically Partitioned Bandwidth for foo and boe Sequences

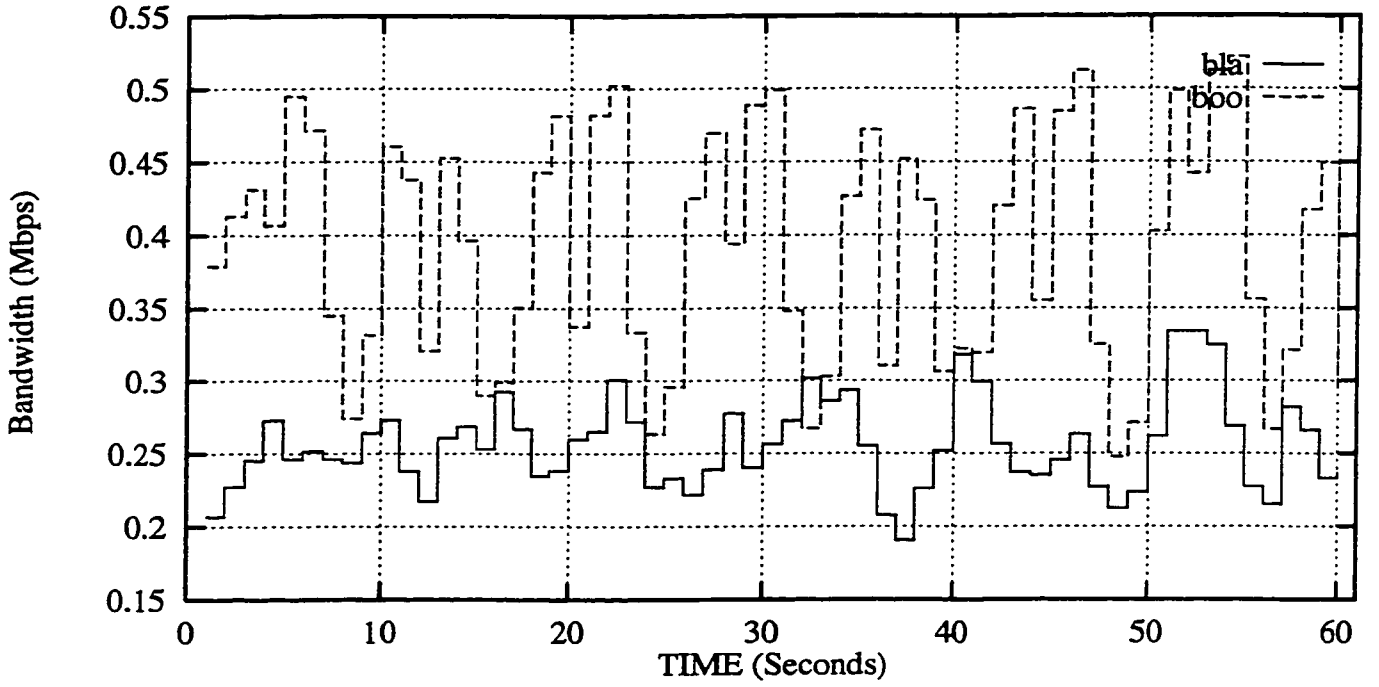


Figure 43: Dynamically Partitioned Bandwidth for `bla` and `boo` Sequences

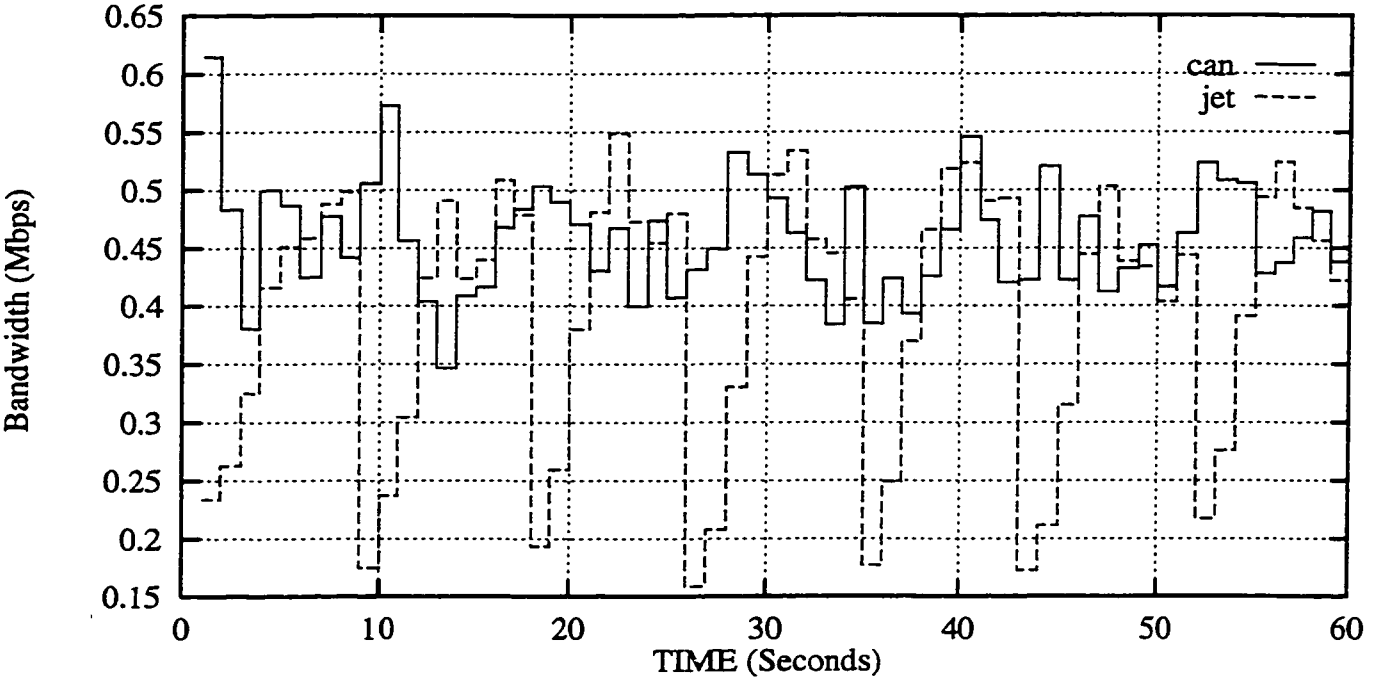


Figure 44: Dynamically Partitioned Bandwidth for `can` and `jet` Sequences

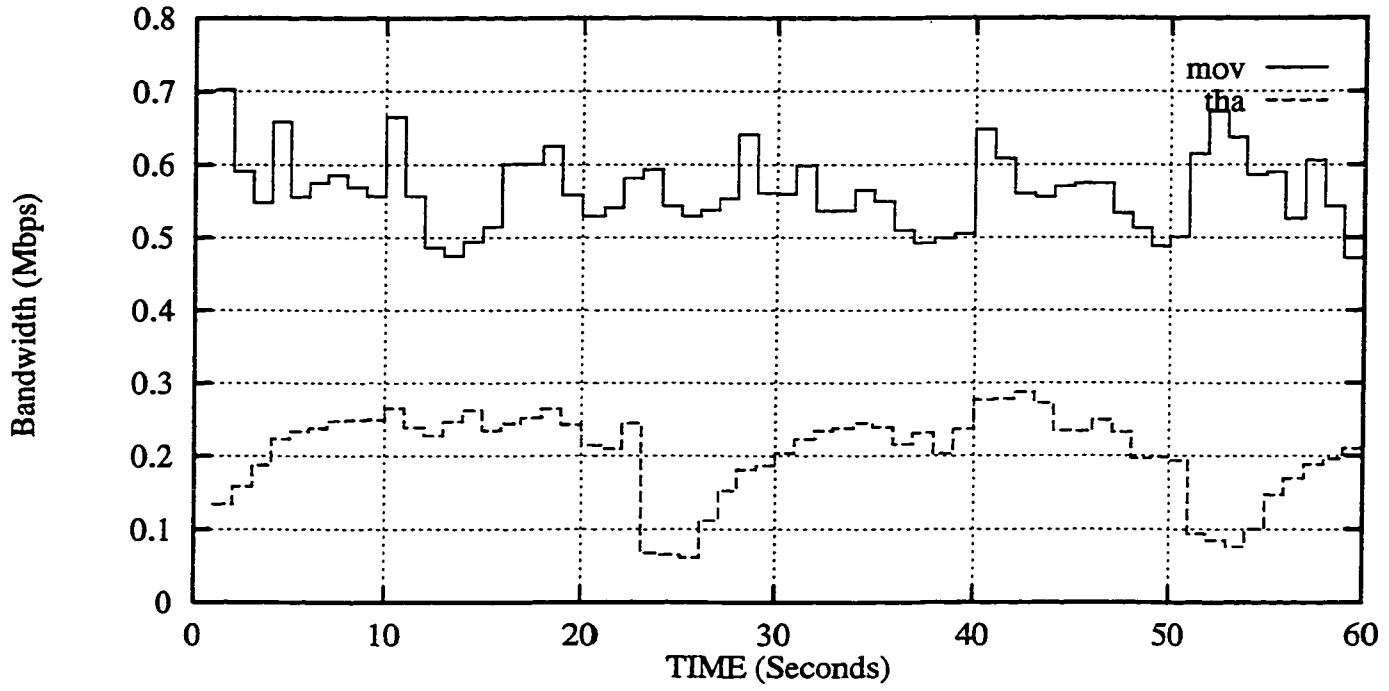


Figure 45: Dynamically Partitioned Bandwidth for `mov` and `tha` Sequences

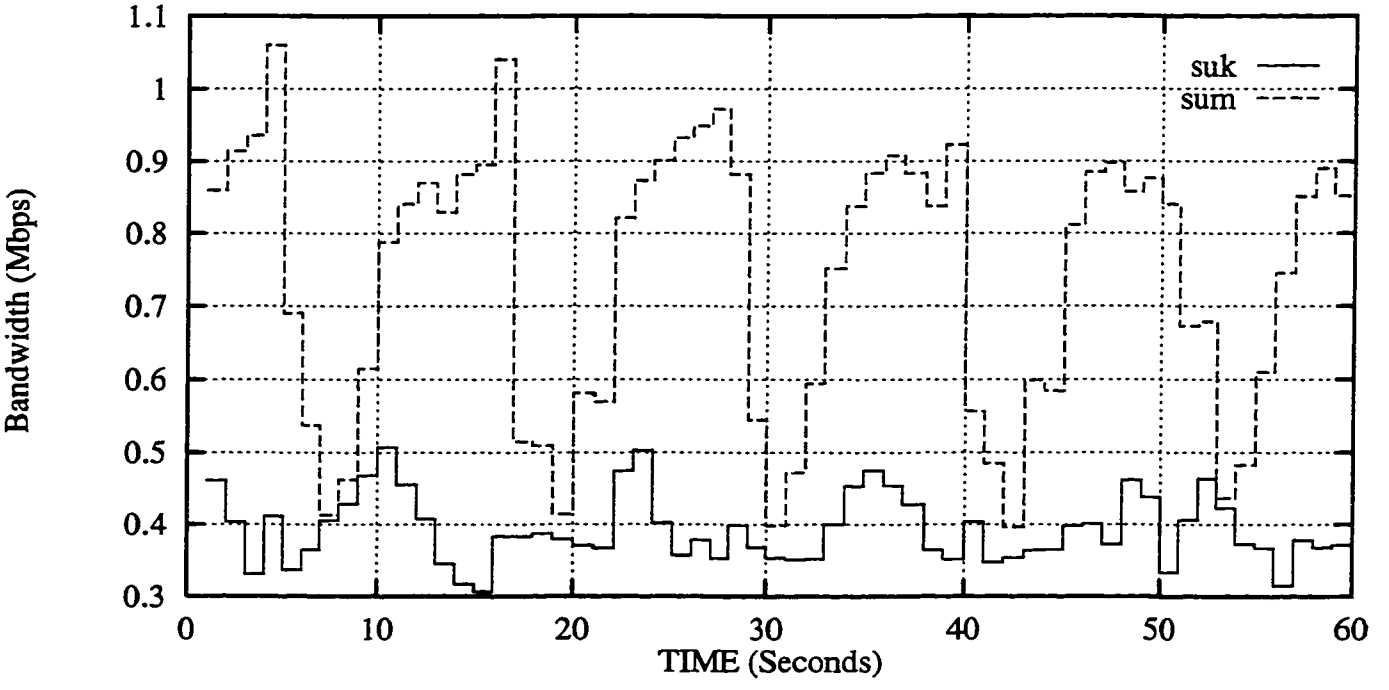


Figure 46: Dynamically Partitioned Bandwidth for `suk` and `sum` Sequences

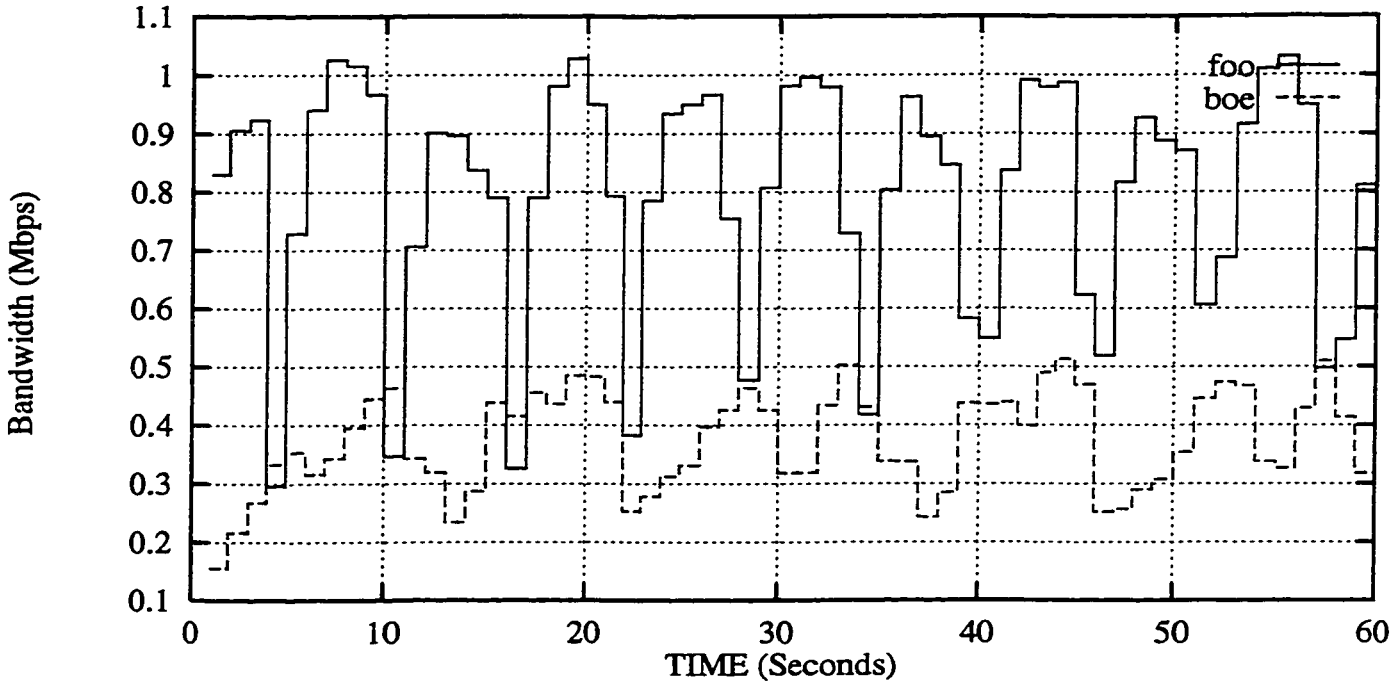


Figure 47: Dynamically Partitioned Bandwidth for `foo` and `boe` Sequences

A.4 Case: 60% *Average Bandwidth*

In this section, we include the dynamically split bandwidth graphs for 60% of *Average Bandwidth* in Figures 48 through 52.

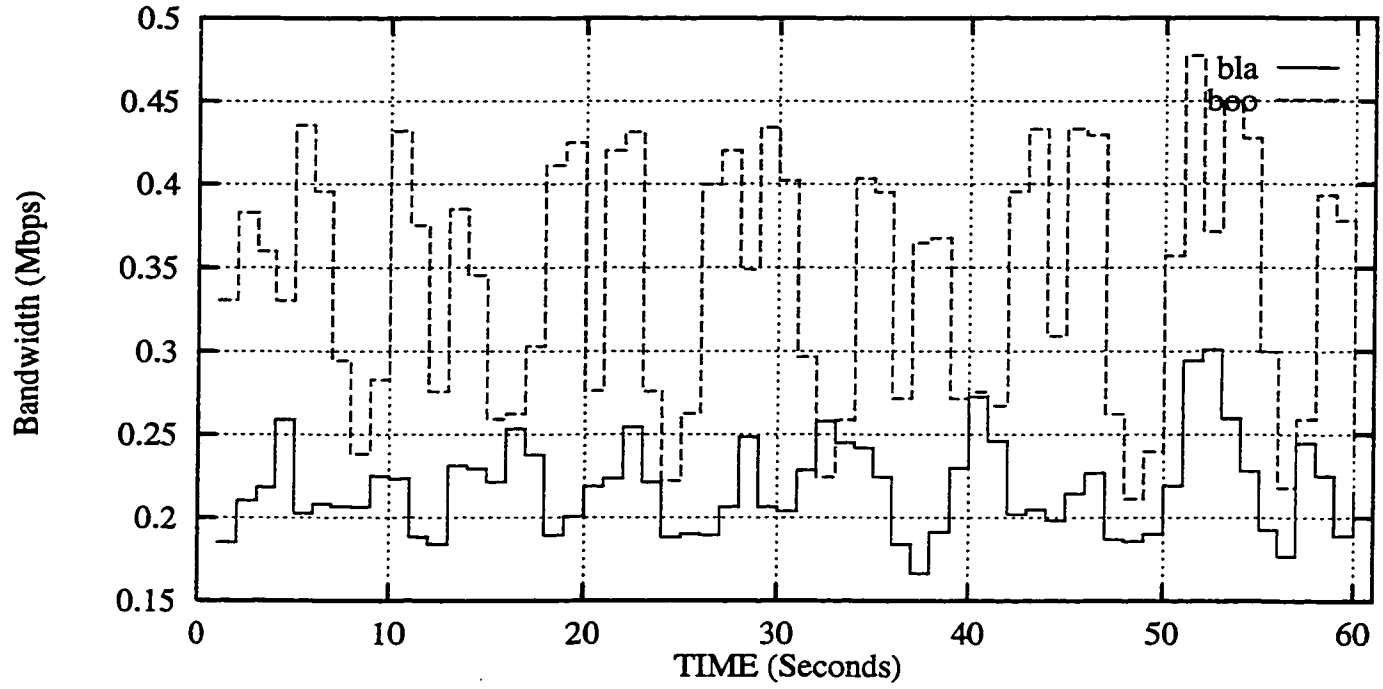


Figure 48: Dynamically Partitioned Bandwidth for `bla` and `boo` Sequences

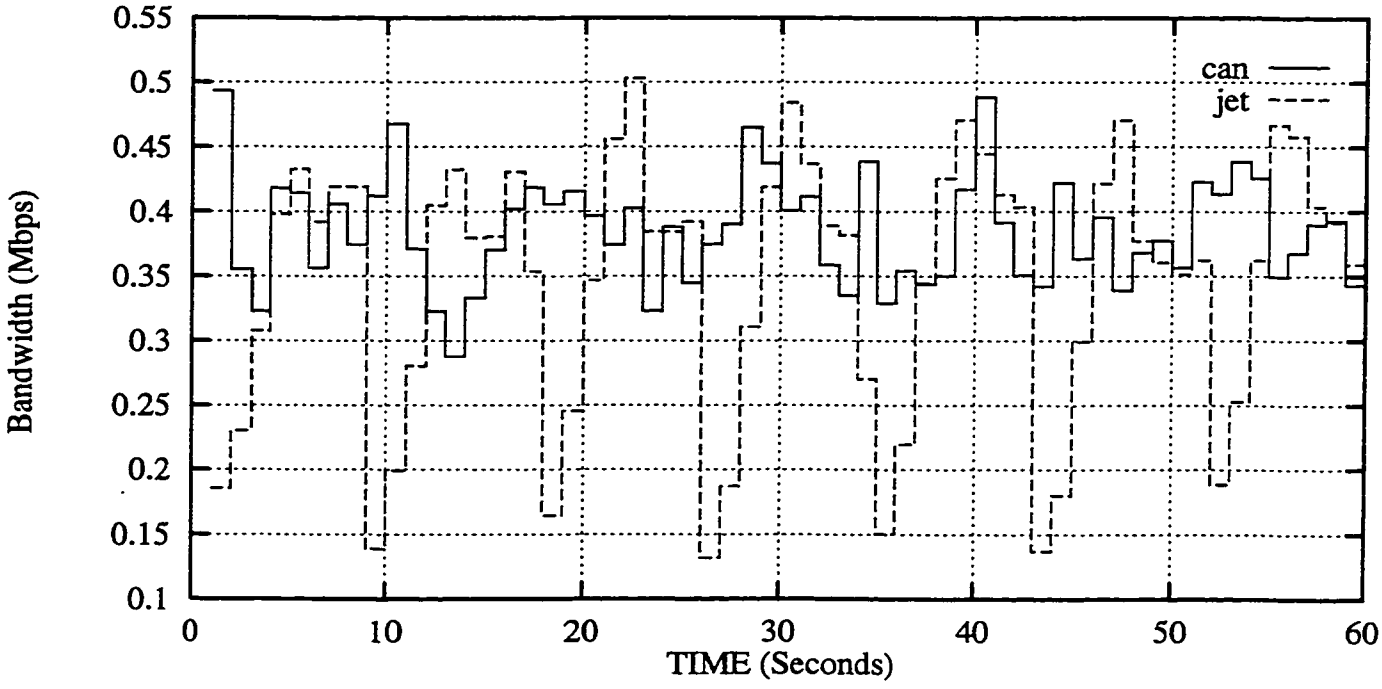


Figure 49: Dynamically Partitioned Bandwidth for `can` and `jet` Sequences

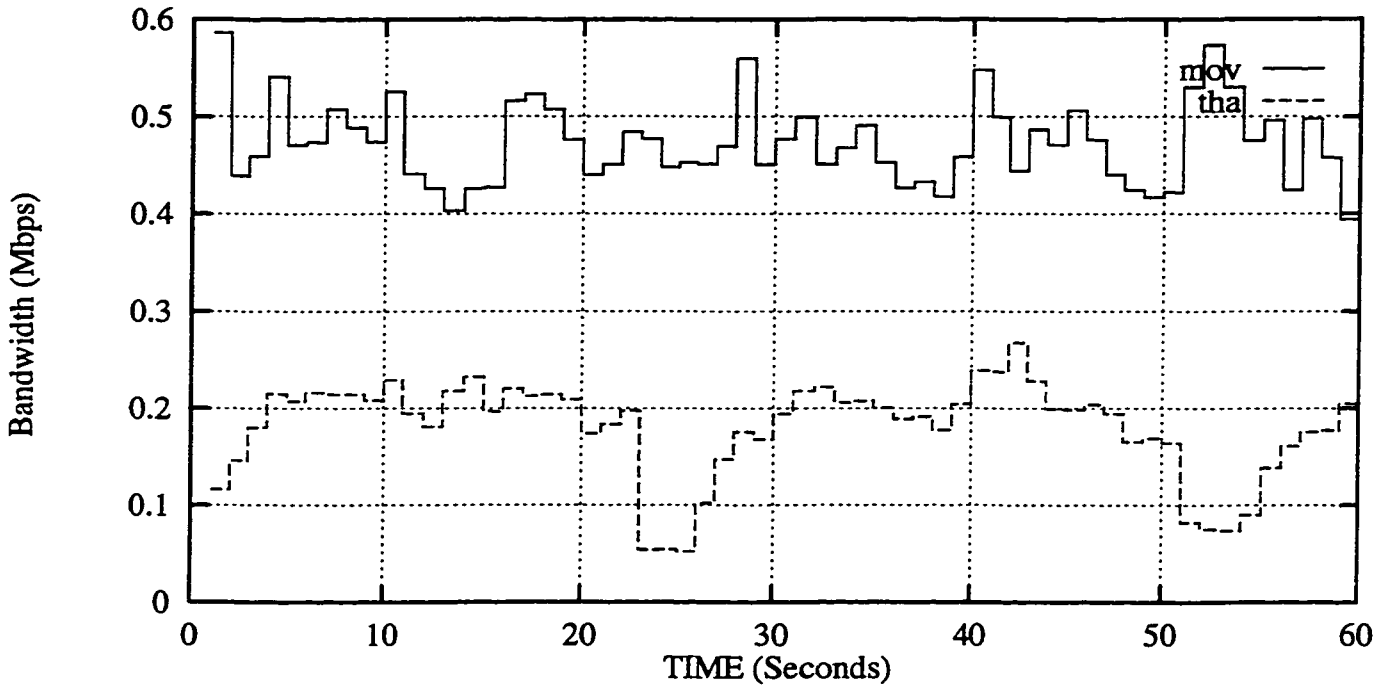


Figure 50: Dynamically Partitioned Bandwidth for mov and tha Sequences

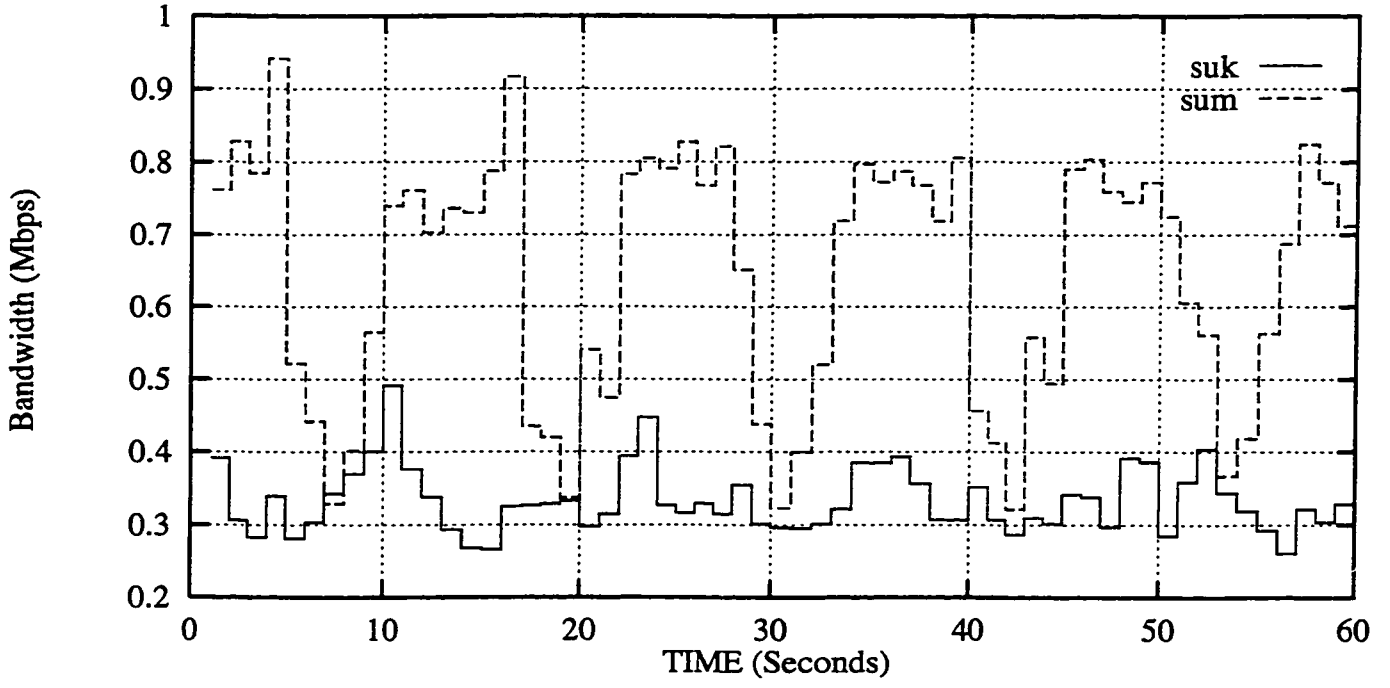


Figure 51: Dynamically Partitioned Bandwidth for suk and sum Sequences

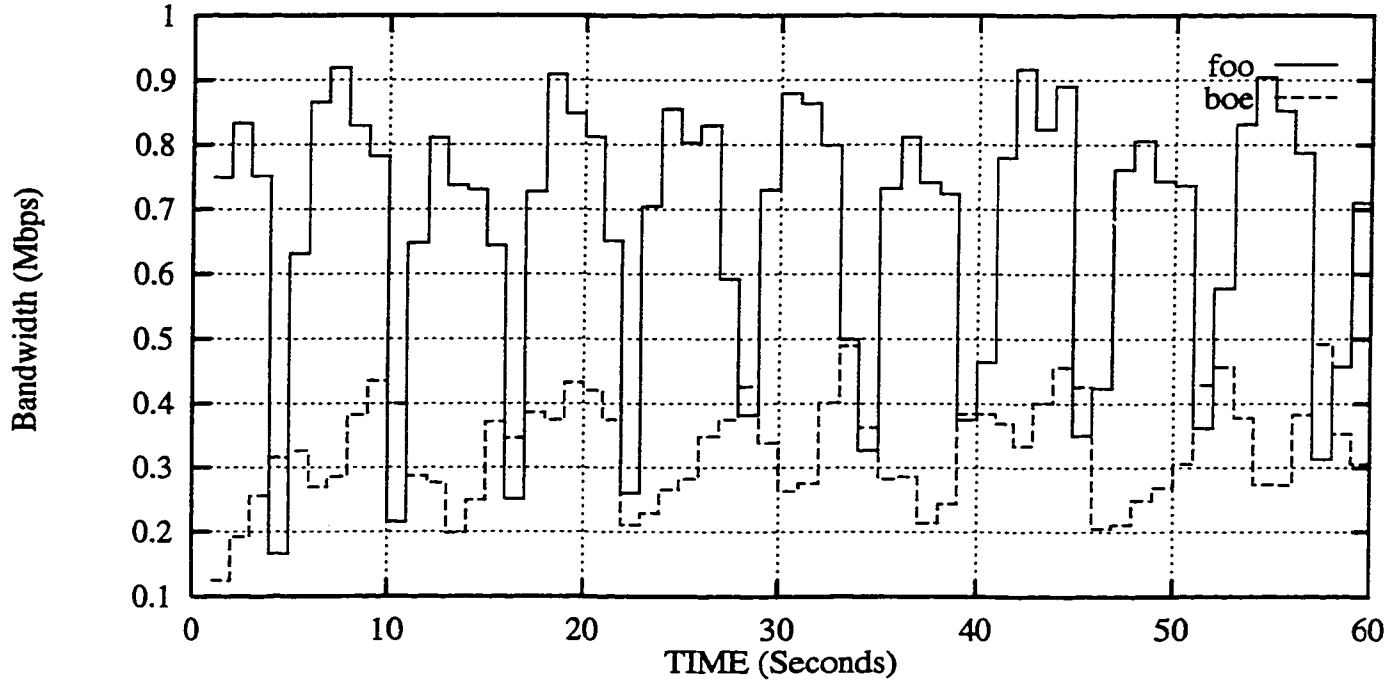


Figure 52: Dynamically Partitioned Bandwidth for `foo` and `boe` Sequences

Bibliography

[11193] ISO/IEC 11172. MPEG Standard, 1993. International Organization for Standardization's Document on Digital Video and Associated Audio.

[AB95] Roalt Aalmoes and Peter Bosch. Overview of Still-picture and Video Compression Standards, Dec 1995. Technical Report No. 95-3, Pegasus Project, University of Cambridge.

[AM95] Elan Amir and Steven McCanne. An Application Level Video Gateway. In *Proceedings of ACM Multimedia*, San Francisco, USA, Nov 1995.

[APKR95] Egil Aarstad, Harald Pettersen, John Kroeze, and Thomas Renger. Experiments on Usage Parameter Control and Connection Admission Control in the EXPLOIT Testbed. In *Workshop on ATM Hot Topics on Traffic and Performance*, Settimo Milanese, Italy, 1995.

[Bat92] Regis J. Bates. *Introduction to T1/T3 Networking.* Artech House, Boston, 1992.

[BCS94] Bob Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture: an Overview, Jun 1994. Request For Comments: 1633.

[BZB+96] Bob Braden, Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource ReSerVation Protocol - Version 1 Functional Specification, Oct 1996. Internet Draft.

[CCGP95] James Cummiskey, Harlan Carvey, Steve Graves, and Dave Petit. IP Over ATM Overview and Tutorial, Dec 1995. Naval Postgraduate School, Working Group's Document.

[Chi95] Leonardo Chiariglione. The Development of an Integrated Audiovisual Coding Standard: MPEG. In *Proceedings of the IEEE*, pages 151–157, Feb 1995.

[Dav92] David D. Clark and Scott Shenker and Lixia Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms. In *Proceedings of SIGCOMM*, pages 14–26, ACM, USA, 1992.

[For95] ATM Forum. ATM User-Network Interface Specification - Version 3.1, Jun 1995. Online Document. Available at: http://cell-relay.indiana.edu/cell-relay/docs/atmforum/ps.html.

[Fur94] Borko Furht. Multimedia Systems: An Overview. *IEEE Multimedia*, pages 47–59, Spring 1994.

[Gal91] Dider Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34, No. 4:47–58, April 1991.

[GB97] Mark W. Garret and Marty Borden. Interoperation of Controlled-Load Service and Guaranteed Service with ATM, Jul 1997. Internet Draft.

[HSS90] Dietmar B. Hehmann, Michael G. Salmony, and Heinrich J. Stuttgen. Transport Services for Multimedia Applications on Broadband Networks. *Computer Communications*, 13, No. 4:197–203, May 1990.

[Kar95] Gunnar Karlson. Asynchronous Transfer of Video. SICS Research Report R95:14, Swedish Institute of Computer Science, 1995.

[Kes97] Srinivasan Keshav. *An Engineering Approach to Computer Networking. ATM Networks, the Internet and the Telephone Network.* Addision Wesley, USA, 1997.

[KM96] Hemant Kanakia and Partho P. Mishra. Packet Video Transport in ATM Networks with Single-bit Feedback. *Multimedia Systems*, 1996:4:370–380, 1996.

[KMS95] Dexter Kozen, Yaron Minsky, and Brian Smith. Efficient Algorithms for Optimal Video Transmission. Technical Report, Cornell University, May 1995.

[KPC97] Rob Koenen, Fernando Pereirs, and Leonardo Chiariglione. MPEG-4 Context and Objectives. In *Image Communication Journal, Special Issue on MPEG-4*, Torino , Italy, 1997.

[LCY94] Simon S. Lam, Simon Chow, and David K. Y. Yau. An Algorithm for Lossless Smoothing of MPEG Video. In *Proceedings of ACM SIGCOMM*, London, U.K., 1994.

[LKPC97] James V. Luciani, Dave Katz, David Piscitello, and Bruce Cole. NBMA Next Hop Resolution Protocol (NHRP), Sep 1997. Internet Draft.

[MF95] Steven McCanne and Sally Floyd. Man Page for the ns Simulator Version 1, Jul 1995. Online Document. Available at: http://www-nrg.ee.lbl.gov/ns/man.html.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, USA, 1994.

[PG93] Abhay K. Parekh and Robert G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integerated Services Network: The Single Node Case. *IEEE/ACM Transactions on Networking*, Jun 1993.

[Poy97] Charles Poynton. Poynton's Color FAQ, Feb 1997. Online Document. Available at: http://www.inforamp.net/ poynton/Poynotn-color.html.

[RRV93] Srinivas Ramanathan, P. Venkat Rangan, and Harrick M. Vin. Frame-Induced Packet Discarding: An Efficient Strategy for Video Networking. In *4th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, U.K., Nov 1993.

[Sat96] Shirish S. Sathaye. ATM Forum, Traffic Management Specification v4.0, April 1996. Ballot Version of the Specification by Fore Systems Inc.

[SCFJ96] Henning Schulzarine, Stephen Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications, Jan 1996. Request For Comments: 1889.

[Sch95] Henning Schulzrine. Internet Services: From Electronic Mail to Real-Time Multimedia. In *Kommunikation in Verteilten Systemen*, pages 21–34. Springler-Verlag, Germany, Feb 1995.

[She95] Scott Shenker. Fundamental Design Issues for the Future Internet, Sept 1995. Internet Draft.

[SJ95] Kai-Yeung Siu and Raj Jain. A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management. *Computer Communication Review, ACM SIGCOMM*, pages 6–20, April 1995.

[SL97] Anastasios Stamoulis and Jorg Liebeherr. S2GPS: Slow-Start Generalized Processor Sharing. Technical Report No. CS-97-03, University of Virginia, Feb 1997.

[SN95] Ralf Steinmetz and Klara Nahrstedt. *Multimedia: Computing, Communications and Applications.* Prentice Hall, USA, 1995.

[SPG97] Scott Shenker, Craig Patridge, and Roch Guerin. Specification of Guaranteed Quality of Service, Feb 1997. Internet Draft.

[Ste93] Ralf Steinmetz. Compression Techniques in Multimedia Systems: A Survey, 1993. Technical Report:43.9307, IBM Networking Center, Germany.

[Ste95] James P. Sterbenz. Protocols for High Speed Networks: Life after ATM? In *Protocols for High Speed Networks IV*, pages 3–18. Chapman and Hall, London, 1995.

[Stu95] Heinrich J. Stuttgen. Network Evolution and Multimedia Communication. *IEEE Multimedia*, pages 42–59, Fall 1995.

[Vet95] Ronald J. Vetter. ATM Concepts, Architecture and Protocols. *Communications of the ACM*, 38, No. 2:30–44, Feb 1995.

[WCS96] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl.* O'Reilly & Associates, Inc., USA, 1996.

[Wor97] John Worclawski. Specification of the Controlled-Load Network Element Service, May 1997. Internet Draft.

[Yas96] Yasser Rasheed and Alberto Leon-Garcia. AAL 1 with FEC for the Transport of CBR MPEG2 Video over ATM Networks. In *IEEE Infocom*, volume 2, pages 529–536, 1996.

[YMGH96] Nicholas Yeadon, Andreas Mauthe, Francisco Garcia, and David Hutchison. QoS Filters: Addressing the Heterogeneity Gap. In *Proceedings of Interactive Multimedia Systems and Services*, Berlin, 1996.

[ZDE+93] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, Sep 1993.