Buffering strategies and Bandwidth renegotiation for MPEG video streams*

by

Nico Schonken

Submitted in fulfillment of the requirements for the degree of

Master of Science

in the Department of Computer Science at

Rhodes University

January 1999

* This research was made possible by the Centre of Excellence at Rhodes University sponsored by Telkom SA Ltd
http://cs.ru.ac.za/coe

Abstract

This paper confirms the existence of short-term and long-term variation of the required bandwidth for MPEG videostreams. We show how the use of a small amount of buffering and GOP grouping can significantly reduce the effect of the short-term variation. By introducing a number of bandwidth renegotiation techniques, which can be applied to MPEG video streams in general, we are able to reduce the effect of long-term variation. These techniques include those that need the a priori knowledge of frame sizes as well as one that can renegotiate dynamically. A costing algorithm has also been introduced in order to compare various proposals against each other.

Table of Contents Page no.

List of Figures Page No.

List of Tables Page No.

Introduction

The goal of this paper is to identify the problem areas within a video-on-demand network and to propose solutions to those problems encountered. The entire video-on-demand system can be broken into three simple components. Firstly there needs to be a source, which we call the video server, from where movies can be requested. The video server has the capability to encode and store available videos and transmit them over a network. We will discuss the MPEG encoding technique in detail in Chapter 1. The network between the source and the destination is the second component of the system. This network can be anything from an ATM LAN to the Internet itself. Lastly, at the destination we have a settop box. This box would usually contain a minimal amount of buffer space as well as the ability to decode the encoded video stream and then display it. Figure A.1. gives an idea of what the system would look like in general.

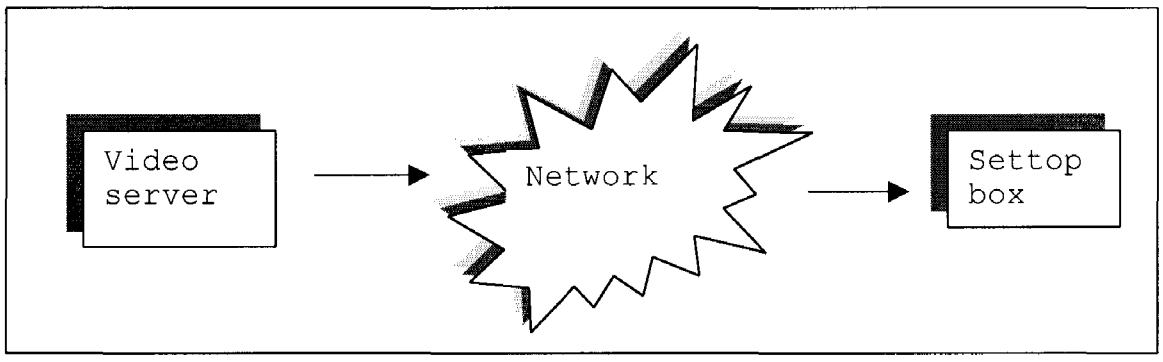

Figure A.1. – A simple illustration of the problem domain

Different problems will occur at various parts of the system. We can thus use various techniques in an attempt to solve each problem. At the video server one can use interval caching methods and read-ahead buffering in order to alleviate burstiness in video servers. This also improves jitter tolerance and hence the real-time performance of the video server. The variable bit rate of MPEG encoded movies causes problems for the video server. The different sizes of the frames cause short-term variation within a videostream. The video server is where decisions for renegotiation of bandwidth would also occur. The server can use the a priori knowledge of the frame sizes in a movie to make decisions about the amount of bandwidth required. Chapters 3, 4, 5 and 6 cover the various methods used to renegotiate bandwidth.

At the 'entrance' to the network, we can use the leaky bucket approach in order to constrain the video stream in such a manner that it conforms to certain parameters. Once again it is the variable bit rate which forces the network to use its resources in order to transmit the video stream at the correct speed.

It is at the settop box where most of our buffering is done. A buffer placed at the destination can alleviate the problems of a variable bit rate. Chapter 2 discusses buffer underflow and overflow. The settop box receives the video

stream at a variable frame rate and must then convert this to a constant frame rate stream in order for the video to be viewable.

Chapter 1 includes a simple overview of MPEG coding in general as well as a more detailed description of the encoding technique and its structure. Some previous uses of MPEG are also included.

Chapter 2 explains what short-term variations occur in an MPEG stream and how one can cope with them using buffering. We introduce the concept of GOP (Group of Picture) buffering and show how the use of this idea can significantly reduce the cost of transporting a video over a network. We use the results of this chapter in most of our experiments in the other chapters.

Chapter 3 introduces long-term variation, which is caused by the changing content within a movie. Chapter 3 gives an overview of the problem and covers the two approaches used to deal with it, namely static renegotiation and dynamic renegotiation. We introduce an objective function for calculating costs of the renegotiation methods. The necessary assumptions are also stated and explained in this chapter.

Chapter 4 deals only with static renegotiation. We have used fixed time lengths in order to slice the movie into intervals with different bandwidth

requirements. Costs are calculated for each of our methods in order to find the most economical method.

Chapter 5 contains a variation of the fixed interval approach. Instead of using equal time lengths to slice the movie into fixed intervals, we have used heuristics and scene changes in the movie in order to make predictions of where bandwidth changes need to be made. We make the implication here that the content of the movie can be a decision tool for points of bandwidth renegotiation. Once again costs are calculated based on the objective function introduced in Chapter 3.

Chapter 6 covers dynamic renegotiation. We use the moving average of the frame sizes and GOP sizes in order to adjust the available bandwidth at run-time. In this case, the a priori knowledge of the frame sizes is not necessary.

Chapter 1

MPEG video compression

1.1. Introduction

The Moving Picture Coding Experts Group (MPEG) was established in January 1988 with the mandate to develop standards for coded representation of moving pictures, audio and their combination and hence it is also the name of the standard that they have produced. [ROSE] suggested that MPEG is expected to cause problems for ATM, since the major part of B-ISDN traffic will be produced by multimedia sources such as teleconferencing and video-on-demand servers and these services tend to use MPEG compression.

1.2. General description

MPEG has been developed for storing video on digital storage media as well as delivering video through local area networks and other telecommunications networks. At a rate of several Mbps, MPEG video is suitable for a range of multimedia applications e.g. video mail, video conferencing, electronic publishing, distance learning and games.

The MPEG standard consists of 3 parts; video encoding, audio encoding and the systems portion, which includes information about the synchronisation

of the audio and video streams. MPEG-1 was *optimised* for synchronised digital video and audio, and compressed to fit into a bandwidth of 1.5 Mbps. The video stream uses up about 1.15 Mbps, while the remaining bandwidth is used by the audio and systems data streams. The system layer contains timing and other information. The MPEG video algorithm can compress video signals to an average of 0.5 to 1 bit per coded pixel. At a compressed data rate of 1.2 Mbits per second, a coded resolution of 352 X 240 at 30 frames per second is often used, and the resulting video quality would be comparable to that of VHS recording. Of course, if we attempted to maintain a high video quality we would then have to vary the required bandwidth. This can be controlled at the encoder by varying the level of compression performed on segments of the video.

MPEG-1 functionalities are a subset of the MPEG-2 ones. MPEG-2 allows for layered coding over ATM where a base layer contains the basic information required, and one or more enhancement layers can be used to improve the quality of the video sequence. The primary application targeted during the MPEG-2 definition process was the all-digital transmission of broadcast TV quality video at coded bitrates between 4 and 9 Mbps. The most significant enhancement over MPEG-1 is the addition of syntax for efficient coding of interlaced video.

When work on MPEG-3 began it originally targeted HDTV (high-definition television) applications with coded bitrates between 20 and 40 Mbps. It was discovered that with some amount of fine-tuning, MPEG-2 and MPEG-1 syntax worked very well for HDTV rate video and thus the work on MPEG-3 was not continued.

When completed, the MPEG-4 standard should enable a whole spectrum of new applications, including videophone, remote-sensing, electronic newspapers, games, interactive multimedia databases and sign language captioning. According to the MPEG-4 overview dated July 1998, this new standard was to be released in October 1998 and would then be an International Standard by December 1998. MPEG has also started work on a new standard known as MPEG-7. It is to be a content representation standard for information search, scheduled for completion in the year 2001.

Full motion video is a set of frames displayed sequentially. Each frame consists of three rectangular matrices representing luminance (Y) and two chrominance values (Cr and Cb) , which hold the colour information. For every four luminance values, there are 2 associated chrominance values: one Cb value and one Cr value.

There are three types of frames in MPEG encoding. Figure 1.1. illustrates these types.

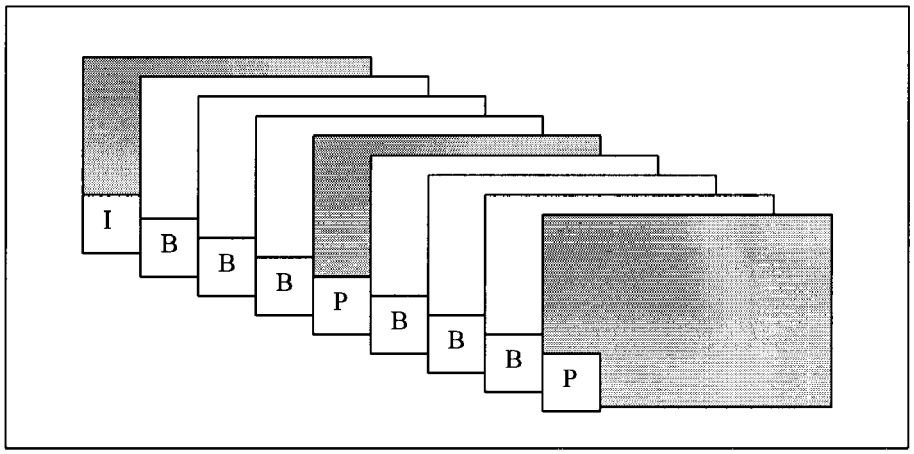

Figure 1.1. – the pattern of MPEG frames (I, B and P frames)

I (Intracoded) frames use only intra-frame coding based on the discrete cosine transform and entropy coding. This means that the I frame is coded independently of any other information in the video stream. A frame with spatial resolution of 640 X 480 pixels and 24 bits per pixel requires about 921 Kbytes to represent when uncompressed. Spatial resolution is the term for the number of pixels that make up the frame. For a video sequence to be displayed at 24 frames per second, the transmission capacity required is about 168.75 Mbps [(640 x 480 x 24 x 24) bits per second] for uncompressed video. Given that MPEG has an optimal compression ratio of 6:1, we can assume that a bandwidth requirement of 168.75 Mbps could be reduced to 28 Mbps.

The discrete cosine transform (DCT) helps separate the image into coefficients of differing frequencies. Low frequency components are more

critical to the human's perception of the image's visual quality. Colour subsampling has been used in MPEG encoding since the human eye is more sensitive to high-resolution luminance than to high-resolution colour. The first compression technique employed by MPEG is to subsample the colour channels in order to reduce them to a quarter of their original sizes. The luminance channel remains its original size while the Cb and Cr levels are reduced to a quarter each and hence when the three channels are put together, the result is a reduction to half of the originally required bandwidth.

Lossy compression is achieved by quantising the resulting coefficients into a smaller set of possible values, with more aggressive quantisation being applied to the higher frequency components. Many of the resulting coefficients may become zero during this process. Entropy coding is a lossless compression technique that further compacts the resulting coefficients, and also takes advantage of runs of zeros. P (Predictive) frames use a similar coding algorithm to I frames, but with the addition of motion compensation with respect to the previous I or P frame. B (Bi-directional) frames are similar to P frames, except that the motion compensation can be with respect to the previous I or P frame, the next I or P frame or an interpolation between them. Interpolation is a method for deducing a value from known higher and lower values. I frames usually require more bits than P frames, while the B frames have the lowest size requirement.

After coding, the frames are arranged in a periodic sequence known as a Group of Pictures (GOP), e.g. IBBPBBPBB IBBPBBPBB etc. The number of frames in a GOP can vary for each video. For a single video, however, all GOP's contain the same number of frames. The sequence of encoded pictures is specified by 2 parameters, M (the distance between I or P frames) and N (the distance between I frames).

e.g. M=3 and N=9 implies IBBPBBPBB IBBP...

M=1 and N-5 implies IPPPP IPPPP IPPPP...

In order to maintain constant quality video, compressed frames are generated at a fixed frame rate (i.e. 24 frames per second), which results in variable bit rate (VBR) traffic.

The output rate of an MPEG encoder depends upon the spatial resolution of pictures (no. of pixels) and the temporal resolution (frequency of sampled data or picture rate), which are parameters typically specified by a multimedia application. The picture rate, as well as some other MPEG encoded parameters can be adaptively controlled to modify the encoder output rate. This output rate changes as the scenes in the video sequence being encoded changes. Frames that are a part of complex scenes require more bits to encode than slower, less complicated scenes. Complex scenes include those with large amounts of motion in them, since it is then that the

P and B frames are heavily affected. It is therefore evident that these scenes require more bandwidth for transmission than less complex scenes.

1.3. MPEG structure

In EBF notation, the MPEG video bit stream structure can be specified as follows:

```
<sequence>              ::=   <sequence header>
                              <group of pictures>
                                    { [ <sequence header> ]
                                    <group of pictures> }
                              <sequence end code>

<group of pictures>     ::=   <group header> <frame>
                              { <frame> }

<frame>                 ::=   <frame header> <slice>
                              { <slice> }

<slice>                 ::=   <slice header> <macroblock>
                              { <macroblock> }
```

The sequence header contains control information needed to decode the MPEG video bit stream. Repeating the sequence header at the beginning of every group of pictures makes it possible to begin decoding at intermediate points in the video sequence. Note that only the first header is required while the others are optional. The group header includes information such as a time code. The frame header contains control information about the frame, such as frame type and temporal reference. The slice header contains control information about the slice, for example, the position of the slice in the frame. Each header, whether it be from a sequence, a group, a frame or

a slice, begins with a 32-bit start code that is unique in the coded bit stream.

The macroblock, however, does not begin with a unique start code. A slice is therefore the smallest unit available to a decoder for resynchronisation and hence important in the handling of errors. If the bitstream contains an error, the decoder can skip to the start of the next slice. Having more slices in a bitstream allows better error concealment, but uses bits which could have been used for an improvement in picture quality. Each macroblock in a slice represents an area of 16 X 16 pixels in a frame. Consider a frame of 640 X 480 pixels. There are 1200 (40 x 30) macroblocks in the frame. By definition, a slice contains a series of one or more macroblocks; the minimum is one macroblock, and the maximum can be all the macroblocks in the frame. Slices in the same frame can have different numbers of macroblocks. Each macroblock header begins with a header containing information on the macroblock address, macroblock type and an optional quantiser scale. Macroblocks are of varying lengths.

1.4. MPEG encoding

In an I frame, every macroblock is intracoded. The technique of spatial compression (intracoding) used to encode I frames in MPEG is essentially the same as that used for encoding JPEG pictures, since I frames do not depend on the P and B frames. These I frames are coded using only

information present in the frame. I frames typically use about 2 bits per coded pixel. This coding algorithm includes using the Discrete Cosine Transform (DCT) in order to transform 8 X 8 blocks of pixels from the spatial domain to the frequency domain. The algorithm then quantises the frequency coefficients. Quantisation maps each frequency coefficient to one of a set of limited number of allowed values. The use of the DCT and quantisation results in many of the frequency coefficients being zero. The coefficients are then organised in a zigzag order to produce long runs of zeros. These coefficients are then converted to a series of run-amplitude pairs, which are then coded with a variable length code.

In P or B frames, a macroblock may either be intracoded, or predicted using various interframe motion compensation techniques. Much of the information in a frame within a video sequence is similar to information in a previous or subsequent frame. The MPEG standard takes advantage of this temporal redundancy by representing some frames in terms of their differences from other (reference) pictures. P frames can propagate coding errors because they are predicted from previous reference (I or P) frames. B frames do not propagate errors because they are never used as a reference. Bi-directional prediction used in B frames decreases the effect of noise by averaging two frames.

This concept of motion compensation, also known as temporal compression, is where MPEG attempts to encode successive frames relative to previous I or P frames. This technique can improve the compression ratio by about 3 times. In temporal compression, the encoder searches forward or backward in time for a similar macroblock. A compressed macroblock contains a spatial vector between the reference macroblock(s) and the macroblock being coded as well as content differences between the two macroblocks. When a macroblock in a P frame cannot be efficiently represented by motion compensation, it is coded in the same way as a macroblock in an I frame. For the B frames, we can possibly have an interpolated match, which is the average of the forward and backward blocks. Since a B frame depends on a reference frame in the future, it cannot be encoded until the reference frame in the video sequence has been captured and digitised. Similarly, a decoder cannot decode a B frame until its reference frame in the future has been received. Therefore, the order in which the frames are transmitted should be different from the order in which a sequence is displayed. The reference frame following a group of B frames in a video sequence should be transmitted ahead of the group.

For example, if the video sequence is as follows,

$$I_0\ B_1\ B_2\ P_3\ B_4\ B_5\ P_6\ B_7\ B_8\ I_9\ B_{10}\ B_{11}\ P_{12}$$

then the transmission sequence is,

$$I_0\ P_3\ B_1\ B_2\ P_6\ B_4\ B_5\ I_9\ B_7\ B_8\ P_{12}\ B_{10}\ B_{11}$$

An MPEG encoder can control its output rate by setting the quantiser scale in the slice header, and also setting the optional quantiser scale in the header of each macroblock within a slice. A coarser setting would result in a lower bit rate at the expense of poorer visual quality. A coarser quantiser scale would discard some of the high-frequency DCT coefficients thereby lowering the encoder output rate. This is due to the assumption that the human eye is less sensitive to such high-frequency information.

1.5. Previous work

[ROSE] found that video sequences such as sports, news and music clips lead to MPEG sequences with high peak bit rate and a high peak-to-mean ratio compared to movie sequences. This results from the rapid movement of a lot of small objects, which increase the amount of data necessary to encode the sequence. [ROSE] also concluded that for modeling of frame and GOP sizes, either histograms, Gamma, or Lognormal probability density functions can be used. [ROSE] found that there are long-range dependencies in the frame sequences although these were difficult to ascertain using the frame-by-frame correlations. [1]

[1] The term correlation is used to describe the degree of linear association between two variables X and Y. When the *coefficient of correlation (r)* has a value close to zero, the data is not linearly related. However when *r* has a value close to +1 or −1, one can assume a positive or negative linear relationship between the two variables. Terms that are correlated over time are said to be autocorrelated.

Other ways to detect long-range dependencies include variance-time plots or periodograms. Our paper has attempted to discuss this problem in more depth and has suggested solutions to alleviate it.

In their paper, [LAM et al] concluded that the rate fluctuations from one frame to the next are most troublesome. Consider an I frame that is 100 kbits in size followed by a B frame 10 kbits in size. If a video application specifies a frame rate of 24 frames per second, sending the I frame over a network for $1/24^{th}$ of a second would require a transmission capacity of 2.34 Mb/s. During the next $1/24^{th}$ of a second, the transmission capacity required for the B frame drops radically to 0.234 Mb/s. These large fluctuations are a consequence of the use of interframe coding techniques in MPEG.

[LAM at el] also presented an algorithm for smoothing frame-to-frame rate fluctuations in a video sequence. The algorithm's performance is improved by a 'lookahead' strategy that makes use of the repeating pattern of I, P and B frames.

[LI et al] introduced the concept of fast playback where only n out of every m frames are displayed (m > n). These frames are displayed at the normal frame rate, but (m-n) frames are not displayed at all. The choice of frames to be dropped determines the nature and quality of the fast playback display.

The nature of the B frames dictates that they can always be dropped without altering the meaning of the rest of the video stream. Each P frame, however, depends on another I or P frame. The video stream will be impacted negatively if these P frames are dropped at random. Given a 12-frame GOP (e.g. $I_1B_2B_3P_4B_5B_6P_7B_8B_9P_{10}B_{11}B_{12}$ $I_{13}B_{14}B_{15}P_{16}$...) we could speed up the playback by a factor of 12 by displaying only the I frames. Dropping just the B frames would increase the playback speed by a factor of 3. It would not be possible, however, to speed up the playback by a factor of 6 by only transmitting I_1P_7 $I_{13}P_{19}$ ..., i.e. only displaying one I frame and one P frame from each GOP. This is because P_7 is predicted and only makes sense relative to P_4, which has been dropped.

In their paper, [KRUNZ et al] suggest a bandwidth allocation scheme for VBR video traffic using the temporal structure of MPEG sources. This scheme results in an effective bandwidth that, in most cases, is less than the source peak rate. This effective bandwidth depends on the arrangement of the multiplexed streams, which is a measure of the degree of synchronisation between the GOP patterns of different streams.

A major challenge in designing B-ISDN / ATM networks is to guarantee the Quality of Service requirements for all transported streams without underutilising the available bandwidth capacity. One could allocate bandwidth based on the peak rates of the sources in order to satisfy the

Quality of Services requirements. This method, however, leads to low utilisation due to the burstiness of many sources (high peak-to-mean ratio).

Statistical multiplexing can be used to increase utilisation by allowing available bandwidth to be shared among various streams. [KRUNZ et al] investigates the bandwidth requirements of video streams that are generated by MPEG encoders. They show that statistical multiplexing can be used to an advantage with MPEG video traffic while providing stringent and deterministic Quality of Service guarantees.

[Doulamis et al] analysed the statistical characteristics of the three types of frames found in an MPEG stream over a B-ISDN / ATM network. They studied the autocovariance and probability density functions of these frames and then proposed two new models that approximate both the statistical properties and the traffic characterisation of MPEG data streams.

1.6. Conclusions

Unlike [ROSE] and [Doulamis et al], this paper does not attempt to find statistical models for MPEG data streams. Although some statistical characteristics are used in this work, we have not presented them as a central theme.

This work is similar to the above mentioned papers in that we have also found long-term as well as short-term variations in the MPEG data. We have only considered a single MPEG stream and have suggested a few bandwidth allocation methods. [KRUNZ et al] on the other hand, have used the multiplexing of video streams as a possible method for allocating bandwidth. Our methods will be described in more detail in the next few chapters.

Chapter 2

Buffering

buffer (n.) a person or thing that lessens shock or protects from damaging impact, circumstances, etc.

a memory device for temporarily storing data. [COLLINS]

2.1. Introduction

In the Information Technology environment, a buffer is a temporary storage mechanism. A buffer can be placed anywhere in a network. It is usually placed near the destination however, because it can be used to smooth variations in a video stream, which makes the displaying of the video much less complicated for the decoder. As data arrives at the buffer it is stored temporarily until the destination is ready to receive it. If the buffer is large enough, the data can be stored for a long period of time. The destination can then receive data at its preferred rate, which is usually very different to the arrival rate. In the context of this study, data would arrive at a variable frame rate while it would generally leave the buffer at a constant frame rate. In this way a buffer can be used to smooth the traffic flow.

It is important to note that a constant bit rate is very good for network transmission because network resources can be allocated optimally, and used to their capacity. On the other hand, a constant frame rate is essential for viewing of the video stream. For example, an encoded MPEG video's normal frame rate is usually about 24 frames per second. Since frames are of different sizes we cannot have a constant frame rate as well as a constant bit rate throughout the movie. The problem of irregular bitrates caused by network latencies aggravates the issue. Frames will travel from a source (video player) across the network, arriving at the buffer at irregular intervals. A buffer placed near the destination can function as a type of adaptor or modifier of the traffic. The buffer smoothes the incoming irregular traffic flow into the more regular 24 frames per second displayable rate. In this way, the video stream is smoothed of almost all variation.

Proposals have also been made where buffers are placed at the entrance of the network. This would not necessarily prevent the data from becoming jittery during its lifetime in the network. Due to network activity and congestion, packets may be lost or slowed down while traversing the network. The transfer of data from the buffer over the network to the destination would not be as smooth as it would if the buffer were placed in

close proximity to the destination. This could mean being placed inside of a set-top box as part of the MPEG decoder/player.

In a video-on-demand system, VCR functions such as stop, pause, rewind and fast-forward may be required. The buffer in the set-top box can also be used to service some of these requests, such as pausing and a short rewind.

2.2. Previous work

For a video-on-demand system, buffering can be used at the server or at the destination, i.e. the set-top box. The amount of buffering (usually measured in Megabytes) which can be used at these points is an important issue.

i) Buffering at the video server

[DAN et al] introduce buffering concepts that alleviate burstiness in video servers. *Read-ahead buffering* is where blocks are read and buffered ahead of the time they are needed. This storing of data improves the on-time data delivery ability of the server with only a small additional cost. This cost includes the cost of the memory for use as a buffer as well as the extra time delay of reading the buffer. Additional read-ahead buffering improves jitter tolerance and hence, the real-time performance of the video server.

This type of buffering cannot, however, prevent jitter and data loss occurring in the network.

Another concept is that of *interval caching* used predominantly in video-on-demand servers where the same data is served to more than one client. Here the data is cached during the intervals between successive streams, which are not synchronised. Closely following streams can exploit the blocks of data that were fetched by the first stream, if memory space is available to retain them. This type of buffering simply reduces the amount of time used to serve videos and the disk traffic within the server, but does not necessarily alleviate the problem of jitter within a video stream.

Buffering is also used to smooth the bitrate fluctuations of encoder output from one frame to another. Without such smoothing, the performance of networks carrying this type of data would be adversely affected. [LAM et al] use the concept of lossless smoothing to eliminate rate fluctuations that are a consequence of interframe coding in MPEG. By placing a buffer at the sender one can smooth out frame-to-frame variations. For example, let the distance between each I frame be 9, let the frame rate be 24 frames per second and let $S_i \ldots S_{i+8}$ be the frame sizes for one GOP. Each frame in that GOP is sent to the network at the same rate, namely,

$$\frac{(S_i + S_{i+1} + \ldots + S_{i+8}) * 24}{9}$$ [average frame size multiplied by frame rate]

Using a buffer at the sender, all frames in that GOP can be sent at the same rate regardless of their individual sizes. The rate of the coded stream still fluctuates from GOP to GOP, due to scene complexity and the amount of motion in a scene. In Figure 2.1 below, the encoder sends out an encoded bit stream. This enters the buffer at the host at a rate equal to the frame rate of the MPEG encoding. Since the frames are different sizes the buffer fills up at a variable rate. The bitrate leaving the host and entering the network is near to constant due to the smoothing effect of the buffer. No data loss occurs in this buffering technique.

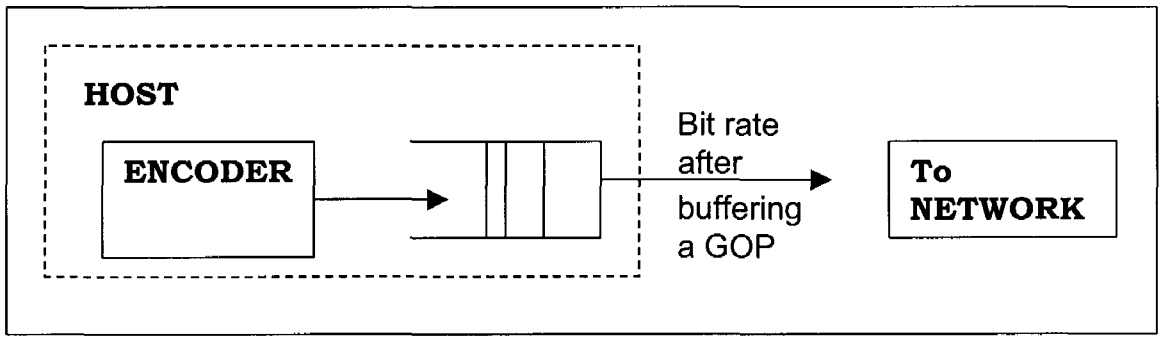

Figure 2.1. – Buffering at sender for lossless smoothing

ii) Buffering at the set-top box

According to [FENG et al], when the buffer size is too small, movies with large variations in frame sizes tend to have large peak bandwidth requirements due to the limited ability of the buffer to smooth large peaks. For small buffers, the peak bandwidth rate is generally based on the

maximum frame sizes, since these large frames cannot be stored as one frame in a buffer and hence they cause large fluctuations in the required bandwidth. For large buffer sizes, the peak rate is based largely on average frame sizes and the long-term burstiness of the video stream. Using large buffers, one can remove most of the burstiness in the stream through prefetching of data. More than one frame can be stored in a large buffer and hence this averages out the required bandwidth.

[LI et al] found that it is advantageous to group MPEG frames into segments. In most research papers, a segment is synonymous with a GOP, but [LI et al] proposed a generalised scheme for grouping frames into segments of any desired length. A frame storage order known as *the minimal causal order* was introduced. This was then used as the basis for grouping frames into segments. This paper also looked at the construction of the MPEG stream itself, paying specific attention to the I, P and B frames in an MPEG video stream. When a B frame is to be displayed, the I or P frame which follows in the playback order needs to be present in the buffer for the successful decoding of the B frame. A buffer storage of at least 2 frames is therefore necessary. It can be shown that the I-frame is either directly, or indirectly, responsible for the successful decoding of a certain number of frames that follows it. According to [LI et al] therefore, when segments are retrieved consecutively for the display of a piece of

video, the first such retrieved segment should be one with an I-frame in it. Note that a GOP will always contain one I frame.

[JAHANIAN et al] used buffering in conjunction with the critical bandwidth allocation algorithm. This algorithm creates a bandwidth allocation plan for video data. The following diagram depicts a possible plan creation.

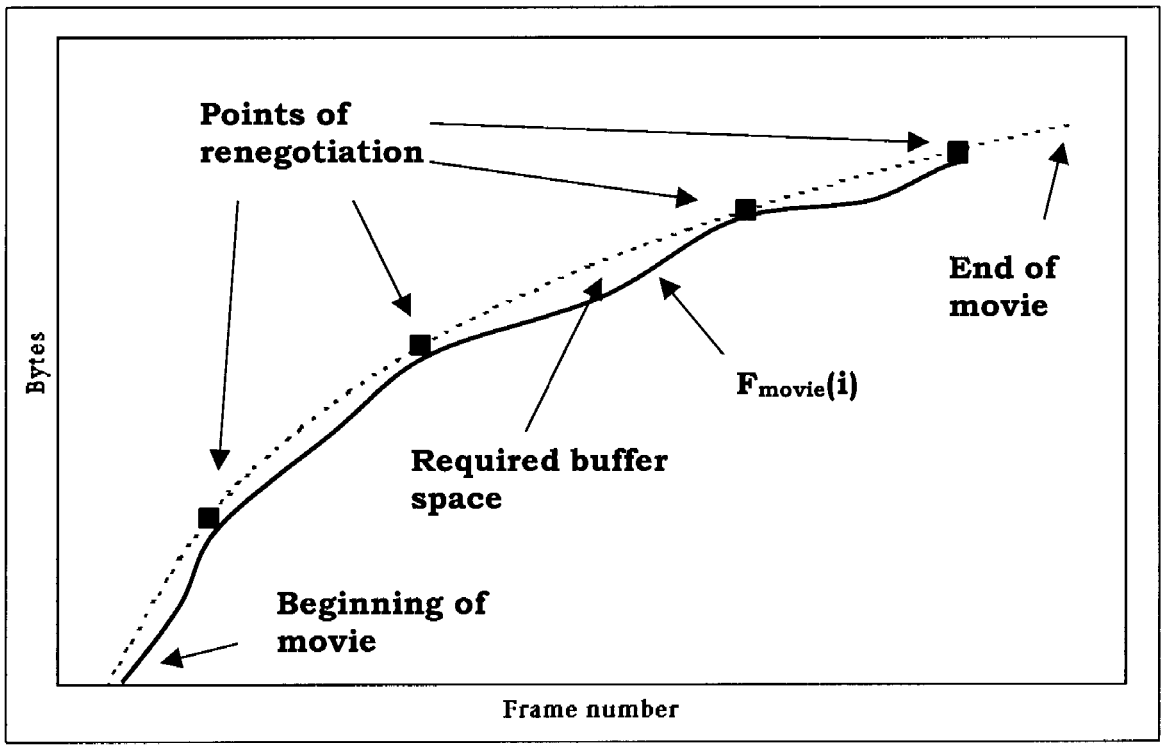

Figure 2.2. – critical bandwidth allocation example

The minimum buffer size required is represented by the maximum vertical distance between the critical bandwidth allocation plan (dotted line) and the function $F_{movie}(i)$ (solid line). The function $F_{movie}(i)$ is given as the cumulative summation of frame sizes for the movie. The four points of

renegotiation show where the amount of required bandwidth was decreased during the streaming of the video. The slope of each dotted line between renegotiation points is the bandwidth requirement for that run.

According to [JAHANIAN et al], the required buffer size is largely due to the long-term burstiness of the video stream, and to a lesser degree, the length and average frame sizes of the videos.

[JAHANIAN et al] also show that for a small amount of client-side buffering one can reduce the number of bandwidth changes necessary. In their experiments, 20 Mbytes of buffer space on the client-side could reduce the total number of changes for the video clips tested, from over 100 changes (in some cases) to less than 10 for all of the videos. Most of the video clips used were full-length movies close to 100 minutes long. It was also shown that if a movie has a sustained area of smaller frames followed by a sustained area of larger frames, the amount of buffering needed tends to be much higher.

2.3. The leaky bucket

The leaky bucket algorithm, discussed in more detail at http://poisson.ecse.rpi.edu/~bergr/BBN/ , is a type of buffering algorithm that is used in ATM networks. It is a buffer mechanism used at the

'entrance' to the network, which constrains the user's data to ensure that it conforms to certain parameters. These parameters include the average bit rate, the peak rate, and the sustained peak time. The buffer is modelled as a bucket of fixed size with a leak. The average bit rate is the maximum rate at which data can leak from the bucket. The peak rate is the maximum rate at which data can be added to the bucket. The sustained peak time is the maximum length of time that peak level traffic may occur. When the client negotiates the required parameters, the size of the bucket will be determined by the need to buffer the peak level traffic for the sustained peak time. Once the bucket overflows, data is discarded. This bucket model provides a policing mechanism, which ensures that the network is isolated from client streams that violate their negotiated contracts.

In the case of an MPEG video stream, the data stream enters the bucket at a variable frame rate. The buffer is then drained at a constant rate of 24 frames per second. The buffer level will always be changing.

In chapter 6 we have used a leaky bucket concept of constantly checking if it is too full or too empty. We use the moving average of the Starwars data in order to do the monitoring. If the bucket gets too full, we increase the leak; while if it gets close to empty, we decrease the rate of the leak. In this

way, the data inside the bucket can be regulated. The following diagram (Figure 2.3.) shows what a leaky bucket regulator does.

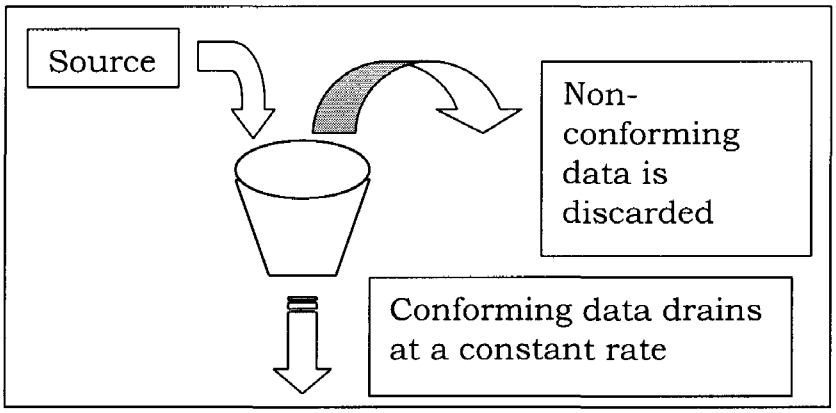

Figure 2.3. – The leaky bucket analogy

2.4. Lookahead

Buffering gives us the ability to smooth a group of frames that are of varying sizes. This involves storing these frames for a short period of time before they are used. Temporarily storing a frame gives the receiver the ability to have some intelligence about the content of that upcoming frame before it is displayed. This effectively introduces lookahead into the stream.

2.4.1. Transmission order

As we know, the I frame in a GOP is the coded still image while the P frames are the deltas, or differences, from the most recent I or P frame. The B frames are interpolations between the I and P frames. We can

therefore say that the frames in a GOP are dependent upon one another. B frames are predicted from the closest two I or P frames, one in the past and one in the future. The sequence of decoded frames ready for display usually looks like this,

I_0 B_1 B_2 P_3 B_4 B_5 P_6 B_7 B_8 P_9 B_{10} B_{11} I_{12} B_{13} B_{14} P_{15} ...

Now, in order for a B frame to be processed (coded and sent or decoded and displayed), the P frame or I frame which is associated with that B frame needs to be processed first. Hence, for the decoder to work, one has to send the associated P or I frame before the B frames. The sending order of frames would then be,

I_0 P_3 B_1 B_2 P_6 B_4 B_5 P_9 B_7 B_8 I_{12} B_{10} B_{11} P_{15} B_{13} B_{14} ...

At the starting point of the MPEG video stream, one would have to decode the I_0 frame, then the P_3 frame and keep them both in memory in order to decode the B frames (B_1 and B_2). One could display the I_0 frame while decoding the P_3 frame, and display the B frames as one decodes them and then display the P_3 frame as one decodes the next P frame (P_6), and so on. This description implies that the buffer should be large enough to hold at least 2 frames while the decoder decodes another. The sender would then be 2 frame-timeslots ahead of the receiver, hence introducing a small amount of lookahead into the system.

2.4.2 Frame size differences

The worst case scenario in MPEG buffering would be where the entire size of a GOP is made up of the I frame (f_0) (which is simply a still picture) and no other frames; i.e. all the other P and B frames are 0 in size. If we assume that the transmission capacity is equal to [GOPsize / 12] per frame-timeslot (for a GOP made up of 12 frames) then the I frame would require all twelve frame-timeslots to download completely. If the I frame is 3 Mbits in size, then the buffer would receive only 0.25 Mbits during each timeslot (one twelfth of its total size). Each twelfth would arrive at time t_0, t_1, t_2 ..., t_{11}, and the I frame (f_0) would only be displayed at t_{12}, when the next I frame is being sent. We would therefore see a delay of at least 12 frame-timeslots, which is equivalent to 1 GOP, between the sender and the receiver. In this case we have a receive latency of up to 12 frame-timeslots.

2.5. GOP Averaging

The largest frame in the Starwars data is 185267 bits in size. If we wanted to carry the entire movie without any buffering, we would require a bandwidth of 4.24 Mbps.

An obvious improvement on the above would be to use GOP smoothing, which implies a small amount of buffering; enough to hold the largest GOP. In our test data the largest GOP is about 0.9 Mbits in size. The

required bandwidth in this case would be 1.80 Mbps. We have therefore already reduced the required bandwidth for the entire Starwars movie by using a small amount of buffering.

In order to compare the effect of using GOP sizes instead of frame sizes we have plotted the bandwidth requirements for the Starwars data set. Using the frame sizes and GOP sizes as input, results in different bandwidth requirements for the movie. In the following graph we have plotted the bandwidth required for every 5 minutes of the movie. We see from this graph that the required bandwidth when using GOP sizes is always below the bandwidth required for the frame sizes. There is a visible amount of smoothing that occurs due to the use of GOP sizes. This implies that the small amount of buffering used within a GOP can smooth the short-term variation in a MPEG video stream.

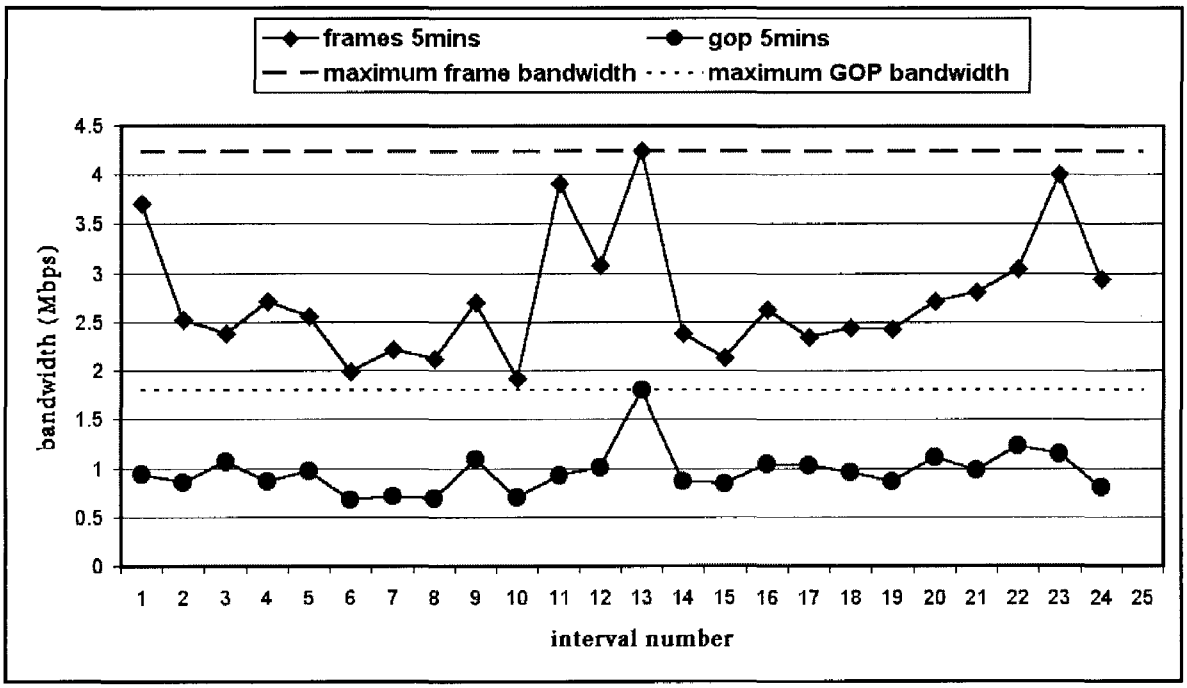

Figure 2.4. – Bandwidth demands for maximum averaged GOPs versus maximum raw frame sizes over 5-minute intervals

In our experiments in chapters 3 to 6, we have used GOP sizes as well as frame sizes as the input data. The differences show the effect of a small amount of buffering used for a MPEG video stream. We have shown that the sizes of different frames within a GOP can be smoothed with the aid of a smallish buffer. A buffer that can hold an entire GOP for the Starwars data will not need to be larger than about 118 Kbytes. Each GOP would be approximately half a second long with respect to the movie time since the frame rate for the Starwars data is 24 frames per second and there are 12 frames in these GOPs.

2.6. Simulator

We have seen that the frame-to-frame or even GOP-to-GOP variations can cause problems for networks and set-top boxes in a video-on-demand system. In section 2.2. we have presented possible solutions to the short-term variation problem, based on related research. In this section we will introduce a tool to assist with finding an adequate buffer size for the MPEG video stream under investigation.

Our initial experiment included constructing a simple simulator, which used the Starwars data from Bellcore, see [GARRETT et al]. Most of the data used throughout this paper originates from traces of a motion picture done at Bellcore. This data is available from [GARRETT et al] as a statistical analysis of a 2-hour long sample of Variable Bit Rate (VBR) video data. The video under investigation, Starwars, contains quite a diverse mixture of material ranging from low complexity or motion scenes to those with very high action. The coding of this data set has been simplified somewhat by the fact that only the luminance component of the video source has been coded. The movie is therefore, in effect, a monochrome video.

Our simulator assumed a buffer of size X (Megabytes) was placed at the receiving end of the network with a given bandwidth of size W (Megabits

per second). The values of X and W were varied while we measured the number of times the buffer underflowed. The bandwidth for each simulation was kept constant throughout the entire run. We did make the assumption that the buffer would not overflow since we would only move data into the buffer if there was space for it. This assumption requires a flow-control algorithm that can stop the sender from delivering data when the buffer is full or near to full. The values of X and W need to be kept within reasonable limits, since buffer size could rapidly increase the cost of the set-top box for the end-user. In our initial experiment, bandwidth values of 0.2, 0.3, 0.4 and 0.5 Megabits per second were used. The buffer sizes used were the values of 0.5, 1, 1.5 and 2 Megabytes. Since there are 174136 frames in the data set, there could be no more than 174136 underflows during the running of the simulation.

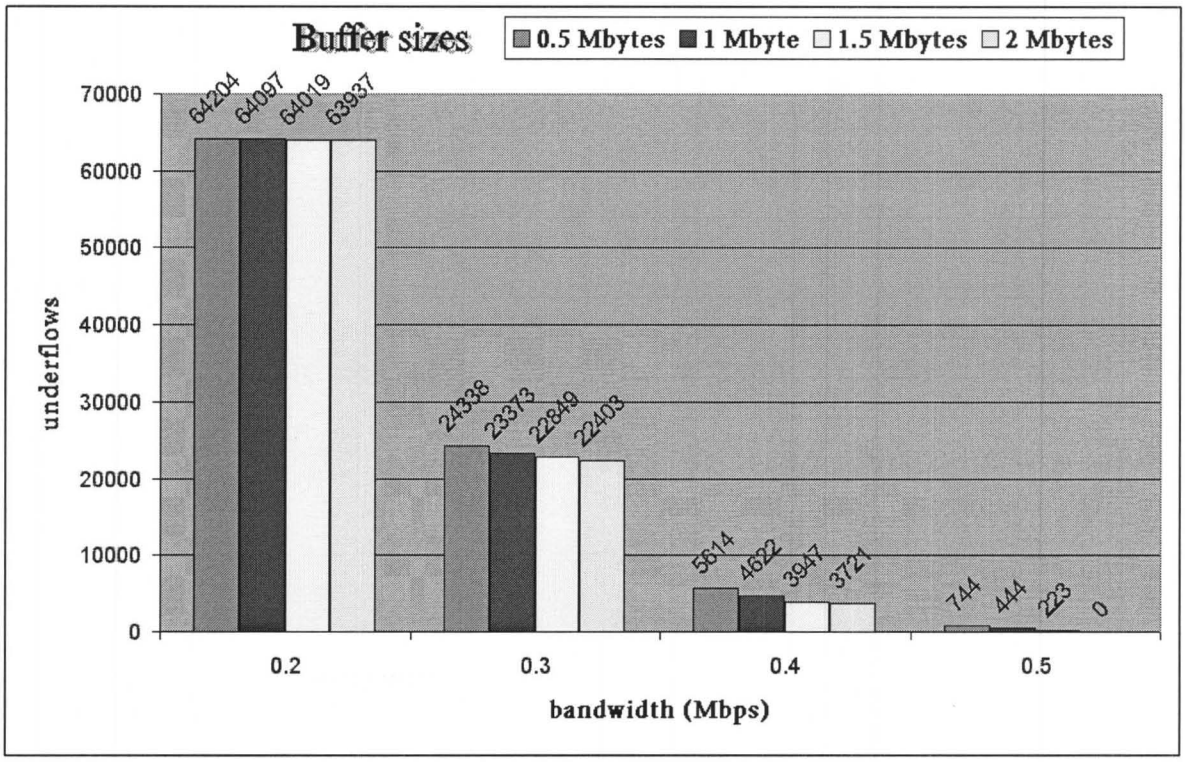

Figure 2.5. – Total underflows vs bandwidth for various buffer sizes and bandwidths

Figure 2.5. shows the different bandwidth sizes used and the respective underflow counts that were generated by the simulator using the Starwars data set. As would be expected, the smaller buffer sizes resulted in a larger number of underflows. As the bandwidth was increased and the buffer size increased, the number of underflows decreased until none occurred with a bandwidth of 0.5 Mbits per second and a buffer size of 2 Megabytes. We see that for a small increase in available bandwidth, the number of underflows is greatly reduced. For example, the difference in the number of underflows decreases substantially with an increase of 0.1 Megabits per second bandwidth from 0.2 to 0.3. One can also note that for the smallest

bandwidth, increasing buffer size to 2 Mbytes does not significantly reduce the number of buffer underflows. According to [GARRETT et al], the average bandwidth of the MPEG video stream under investigation is approximately 0.36 Megabits per second.

The size of the buffer requirements will increase as the available bandwidth decreases. For example, at a bandwidth of 0.4 Mbps we require a 12 Mbyte buffer for 0 underflows, while a bandwidth of 0.3 Mbps (below average) requires 52 Mbytes for 0 underflows. In our simulation, we are only interested in steady-state behaviour, hence we will not cover the case when the available bandwidth is below the bandwidth average for the entire movie.

[GARRETT et al] also show that the peak-to-mean ratio for the Starwars data is 11.88. This ratio is a primary indicator of the variability of the stream. In the case of our data set, the peak frame size is almost 12 times the size of the average frame size. The range of the data is also quite large, which adds to the bursty nature of the video stream. The maximum frame size is 185267 bits, while the minimum frame size is only 476 bits.

Using this simulator along with the data and analysis of [GARRETT et al], we have attempted to determine to what extent the strategic placement of

one or more buffers in a network can smooth multimedia traffic; in this case MPEG video streams. If there are too many occurrences of frame loss or underflows then adjustments to the approach need to be made. We have, however, shown that a buffer of 2 MB is efficient for smoothing our MPEG data set when used in conjunction with a bandwidth of 0.5 Mbps, since the buffer does not underflow at this rate.

Note that any bandwidth requirement is bounded on the upper and lower side. The upper bound would be the peak frame size. It would be unnecessary to allocate more than the maximum frame size as a bandwidth requirement. The lower bound would then be the average frame size.

In the above experiment, our simulator has used individual frame sizes as input. In essence we have used the bandwidth required for each frame. Therefore, large bandwidth requirements do not last for long periods, but rather only for a few seconds.

2.7. Conclusions

In this chapter we have shown how buffering at the set-top box can assist in coping with the short-term variation found in MPEG encoded data. Buffering a variable bit rate datastream can reduce the peak bandwidth

requirements of a MPEG video stream. We have also explored the tradeoff between buffer size and average bandwidth. We have shown how the use of a large enough bandwidth and buffer size can reduce the number of buffer underflows during the transmission of a movie over a network.

We have given some descriptions of previous research where small amounts of buffering and caching have been used at the source of the video stream. The leaky bucket approach to buffering, used at the entrance to a network, has also been introduced in this chapter. We have seen how this type of buffer can be used to regulate conforming and non-conforming data.

A buffer can be placed anywhere in the network, not necessarily just at the sender or just at the receiver. It must be noted however that placing the buffer at the entrance to a network will not be able to prevent the jittery effect caused by a congested network. Buffering can also be useful for lookahead in a data stream. In rest of this paper, we have investigated mechanisms for dealing with the possible long-term variations in MPEG video streams.

It has been shown in this chapter that grouping frames into a GOP has the effect of smoothing the data. If we have enough buffer space to hold an

entire GOP (half a second of movie time), then the required bandwidth is reduced substantially. With this result in mind, we can approach the following chapters using GOP data as the input to most of our experiments.

Not only does buffering benefit us by reducing the required bandwidth for a movie, but it can also be used to smooth any jitter, which is caused by network latency.

Chapter 3

Renegotiation

3.1. Introduction

MPEG video traffic is inherently bursty, especially in the short-term, and it also exhibits long-term variations with respect to bandwidth requirements. This short-term variation is largely due to the different frame sizes which occur in MPEG encoded streams. The short-term burstiness can be dealt with by using buffering techniques as shown in Chapter 2.

Long-term variation is due to the gradual increase or decrease of the average frame size during a movie. When the content of a movie is complex, the average size of the frames will increase. This will happen rapidly when there is a change from one scene containing very little action to one where there is a large amount of complexity and motion. There will then be a gradual decrease in the average frame size, as the scenes become less complex. These gradual increases and decreases of frame sizes lead to long-term variation in the movie. Buffering will not be able to reduce this type of variation since long-term variation occurs gradually, over a large number of frames, while buffers are generally chosen to handle a small number of frames. We will attempt to manage these types of fluctuations by

renegotiating the available bandwidth at intervals during the movie in order to increase or decrease the required bandwidth when necessary.

3.2. The problem

In order to show that long-term variation does exist in our test data we need to compare the frame sizes in Table 3.1. to those in Table 3.2. There is a large difference in the average frame sizes of the two tables. To show that this difference is statistically significant, we have tested the hypothesis:

$H_o : \mu_2 > \mu_1$ vs $H_a : \mu_2 = \mu_1$ (μ_i refers to the frames in Table 3.i)

Here μ_1 and μ_2 are the mean frame sizes from different sections of the Starwars movie. According to our test, there is a definite statistical difference between the average frame sizes within the movie. This difference shows that there is a long-term variation of frame sizes in this video stream, which can cause bandwidth allocation problems. The rest of this chapter suggests ways to counter this problem.

Type	Size
I	12609
B	7022
B	11091
P	10960
B	3053
B	11531
P	11159
B	2357
B	11440
P	11647
B	11911
B	11398
Average frame size (μ_1)	**9681.5**

Table 3.1. – Example of frame sizes near the beginning of the Starwars data

Type	Size
I	49150
B	23668
B	24465
P	46802
B	24863
B	24311
P	44832
B	22666
B	23880
P	44227
B	23508
B	24676
Average frame size (μ_2)	**31420.67**

Table 3.2. – Frame sizes from the middle of the Starwars data

3.3. Previous work

[NORIAKI et al] have proposed a dynamic bandwidth allocation method for an interactive video-on-demand system. Required bandwidth is determined based on the queue length at the set-top box. No pre-calculation is necessary for this scheme hence it is beneficial for interactive video-on-demand. Used in conjunction with a buffer at the set-top box, the bandwidth is increased when an underflow is predicted and decreased when an overflow is predicted. This philosophy is similar to that of the "leaky bucket" algorithm mentioned in Chapter 2. In general, either the set-top box can monitor its own buffer (but then some sort of signalling information would be required so that the set-top box can inform the sender when to increase or decrease the traffic flow) or the sender, given knowledge of the set-top box buffer size, can model what will be happening at the remote end, and adjust the bandwidth on that basis.

[JAHAN et al] introduced an optimal bandwidth allocation (OBA) algorithm that minimised the total number of bandwidth changes necessary for continuous playback. The OBA algorithm is a variant of the critical bandwidth allocation (CBA) algorithm.

For a fixed buffer constraint, the CBA technique results in plans for continuous playback of stored video that have (1) the minimum number of bandwidth increases, (2) the smallest peak bandwidth requirements, and (3)

the largest minimum bandwidth requirements. The OBA algorithm considers a more complex case where, in addition to the three critical bandwidth allocation properties, it minimises the total number of bandwidth changes necessary for continuous playback.

3.4. Our approach

We have investigated the delivery of MPEG video streams over negotiated-bandwidth channels. A number of methods for adaptively renegotiating the delivery bandwidth have been considered. In each method, a bandwidth is assigned at the start of the video stream and it can then be adjusted (renegotiated) according to the needs of the video by whatever technique is used in the given method.

Using the a priori knowledge of stored MPEG video data (the frame sizes) we have developed techniques which allow us to know at the start of the video stream what the bandwidth requirements for the entire duration of the movie will be. By pre-analysing a movie and comparing scene and group of picture sizes we are able to detect the points in a movie at which to adjust the available bandwidth in order to reduce the effect of long-term variation in MPEG video streams. Since not all video servers will have the a priori knowledge of frame sizes for the movies they are serving, we have also experimented with a dynamic renegotiation approach. Using the moving average statistic we are able to dynamically adjust the required bandwidth.

The two basic strategies covered in chapters 4, 5 and 6, with respect to bandwidth renegotiation, are therefore the pre-analysis approach and the dynamic analysis approach. The pre-analysis approach calculates the bandwidth required for known interval lengths. These intervals may be of a fixed size throughout the movie or they may be of varying sizes. Chapter 4 deals only with the fixed interval scenario. Chapter 5 uses more flexible methods for choosing interval sizes, and considers heuristic methods in order to determine possible renegotiation points. The scene changes within the movie are also considered as possible sites for renegotiation.

The dynamic analysis approach, discussed in Chapter 6, calculates the amount of bandwidth required without the use of pre-determined interval lengths. Using a moving average, we renegotiate for an increase or decrease in required bandwidth based on upper and lower bounds that change throughout the length of the video stream.

In order to compare the results from the above experiments we will need a base measurement against which they can be evaluated. We have therefore calculated the minimum bandwidth necessary to carry the entire movie without renegotiation and then use that as a basis, against which, any cost savings accrued during renegotiation, can be measured. Since we are using one bandwidth value throughout the entire datastream, there will only be a single negotiation of bandwidth at the start of the movie.

3.5. Assumptions

For our experiments, we assume that the network has adequate bandwidth available for use by the MPEG video streams at all times. In this way, the cost will not be affected by a lack of bandwidth. Furthermore, we assume a linear cost function, i.e. doubling the bandwidth requirement would double the cost. This implies that the cost of carrying K Megabits is independent of the bandwidth, i.e. 100 Mbps for 1 second costs the same as 1 Mbps for 100 seconds. The unit of costing is therefore taken as Megabits, i.e. one Mbps for 1 second would incur one unit of cost, while 10Mbps for 60 seconds would cost 600 units. We assume the objective function for the minimisation problem must minimise these unit costs. The costs involved would include the amount of bandwidth actually used over each time period as well as the cost of renegotiating between a high and a low bandwidth. This renegotiation cost is also assumed to be a constant cost incurred every time we renegotiate and can also be expressed in our units.

Cost = Σ(no. of seconds * bandwidth) + (no. of renegotiations * K)

where K = renegotiation cost

This costing function is calculated as follows. We sum up the amount of bandwidth used over the length of time that the specified bandwidth was used. Whenever the bandwidth is adjusted there is a renegotiation cost equal to K. Finally, we assume that the cost of a certain bandwidth over a

fixed interval will be fixed, and will not depend on external factors such as the current network load.

Given the above costing algorithm, we are now able to calculate the cost of the minimum bandwidth required to transmit the entire video stream, using only one negotiation of bandwidth at the very beginning of the transmission. We would obtain a cost of approximately (30767 + K) units when using the frame sizes of the movie as input, and a cost of (13080 + K) units when using the GOP sizes. The small amount of buffering implied when using GOP sizes instead of frame sizes reduces the cost of transporting the movie, which is consistent with our findings in Chapter 2. We have used these costs as a basis against which we will compare the other renegotiation strategies in Chapters 4, 5 and 6.

3.6. Conclusions

This chapter has identified the problem of long-term variation. Its presence in the Starwars data has been verified. The size of any frame is dependent on the content of the movie at that point.

In order to reduce the negative effect of this long-term variation, we have introduced the concept of renegotiation in this chapter. We have mentioned a number of methods that can be used for adaptively renegotiating the delivery bandwidth for an MPEG video stream. These methods include using

fixed intervals at which to renegotiate as well as using heuristics to determine various interval lengths. A dynamic method of using the moving average of the data set will also be discussed in detail in Chapter 6. A simple model against which we will compare the costs has been proposed.

Chapter 4

Fixed Intervals

4.1. Introduction

Chapter 4 deals only with the renegotiation of bandwidth over fixed intervals. Each interval will be of the same length with respect to time, but different bandwidths will be required to transport the data for each interval since the amount of data per interval will vary. The number of intervals chosen will have a direct effect on the overall cost of the transmission of the movie. This is due to the fact that each interval implies a new bandwidth, which implies another renegotiation would be required. Assuming a non-zero renegotiation cost, a large number of intervals would increase the total cost.

Based on our results from Chapter 2, we only need to use the GOP sizes in our experiments. We have seen how the use of buffering can significantly reduce the bandwidth required to transport an entire movie.

4.2. Pre-analysis using Fixed Intervals

Fixed-interval renegotiation simply splits the video stream into fixed intervals and renegotiates the required bandwidth at the start of each interval. The bandwidth requirement will depend on the GOP sizes within

each interval. Rather than attempting to negotiate for the maximum bandwidth for the entire video stream, the video server will need to negotiate for a different, and often reduced, bandwidth requirement for each interval. Clearly, renegotiating the bandwidth for each GOP will minimise the bandwidth requirements, but because of the reasonable assumption that there are non-zero renegotiation costs, this method will not necessarily return the least cost. We wish to choose the interval size in such a way that the total cost will be minimised.

We have divided the movie into intervals, 1 GOP (half a second), 7.5 seconds, 15 seconds, 30 seconds, 1 minute, 2 minutes, 5 minutes and 10 minutes. The bandwidth for each interval was calculated using the maximum GOP size for that particular interval. For example, if the largest GOP in an interval of X minutes was 500 000 bits, the required bandwidth for that interval would then be 0.95 Mbps. The bandwidth would be maintained for X minutes of the movie before a renegotiation would take place at the beginning of the next X-minute interval. This method would only be possible if the GOP data were available prior to the streaming of the movie, since the server would need to know what the largest GOP for each interval would be before negotiating a bandwidth for that interval.

In Figure 4.1. we have plotted the upper and lower boundary for the required bandwidth for the Starwars data. The bandwidth required for

each GOP has been plotted against the maximum bandwidth required if only one bandwidth was set (implying no negotiations of bandwidth).

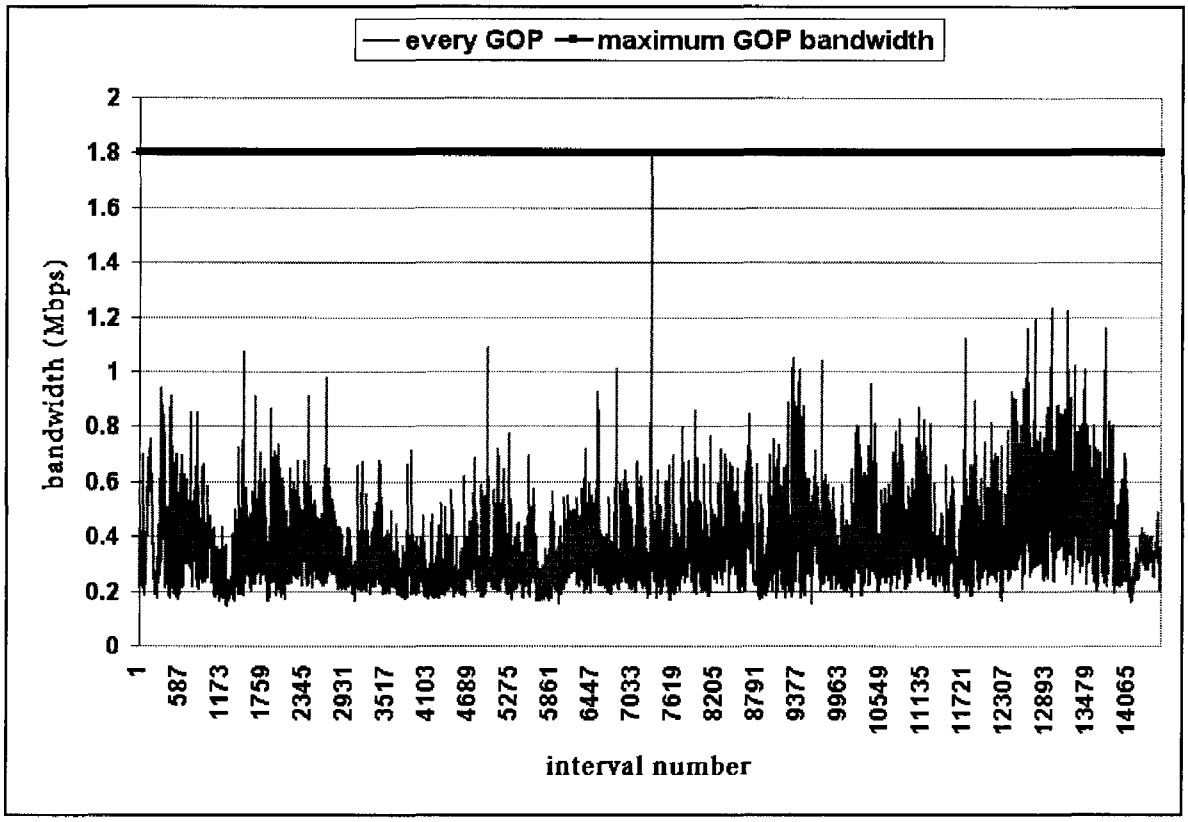

Figure 4.1. – Bandwidth for every GOP vs bandwidth for largest GOP

In Figures 4.2. and 4.3., we have plotted the required bandwidth for 2-minute and 5-minute intervals respectively, using the GOP sizes as the input data. For our data, 1.80 Mbps is the maximum bandwidth required if there were to be no renegotiations during the movie. We see here that a bandwidth of 1.80 Mbps is not necessary to maintain throughout the entire movie. In the following two graphs, 1.80 Mbps has been plotted as a

comparison. We have also plotted the average bandwidth required if we renegotiated after every GOP.

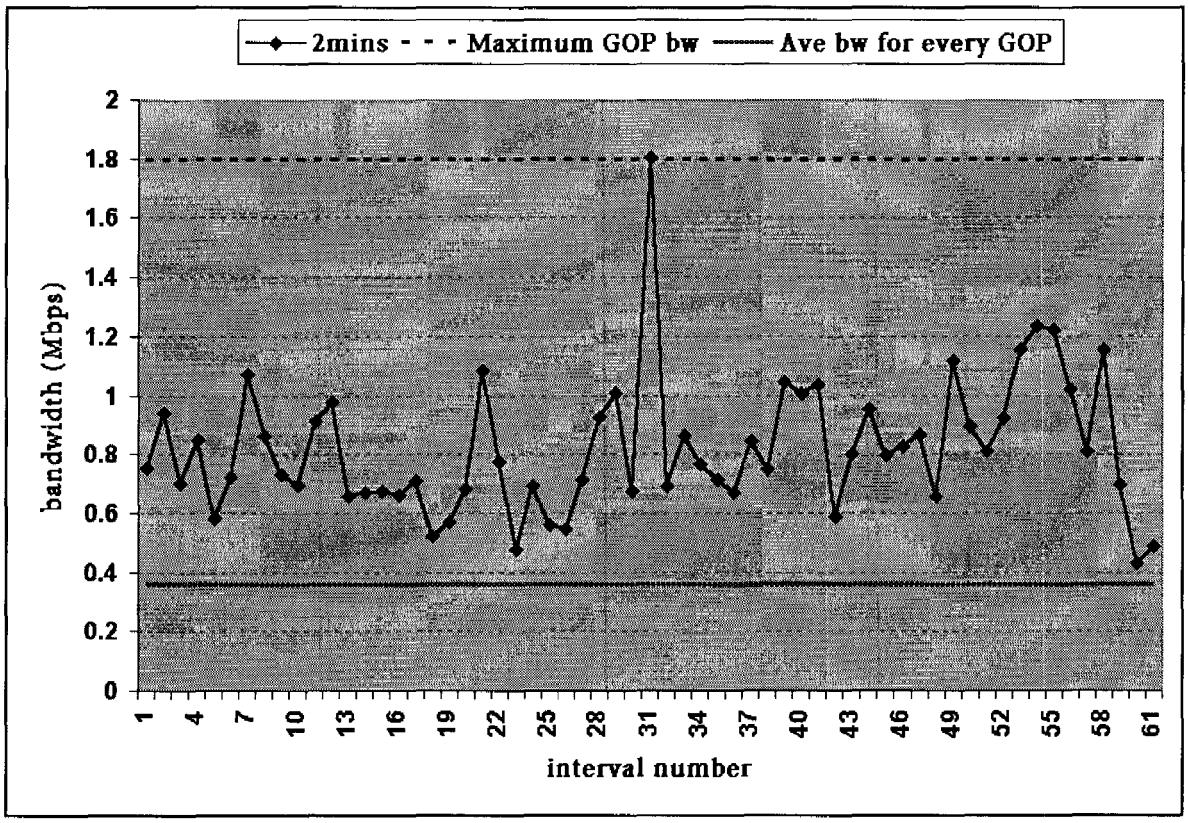

Figure 4.2. – Bandwidth requirements for all 2-minute intervals (GOP sizes as input)

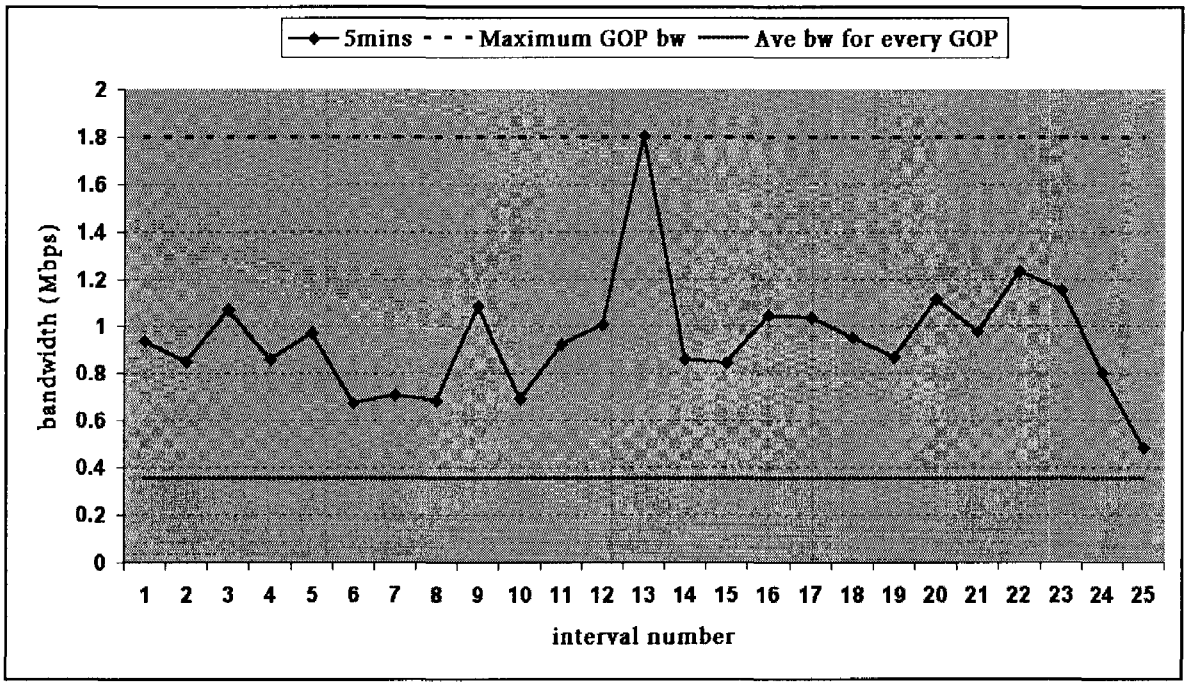

Figure 4.3. – Bandwidth requirements for all 5-minute intervals (GOP sizes as input)

Immediately we notice the large number of renegotiations of bandwidth that occur when the movie is divided into 2-minute intervals. There are 61 renegotiations in Figure 4.2. as opposed to only 25 in Figure 4.3. Both graphs show a maximum bandwidth requirement of 1.80 Mbps. However, the average bandwidth requirement for all 2-minute intervals is 0.82 Mbps whereas the average for all 5-minute intervals is 0.946 Mbps. Over a period of one hour, this equates to a difference of 56 Mbytes. Notice also the large area between the lowest graph line and the constant 1.8 Mbps line. This is the effective 'wasted bandwidth' if we cannot renegotiate. It is this area that we seek to minimise.

It may be of interest to the reader to note that the peaks in the middle of the two previous graphs are due to scenes in the Starwars movie, which contain large amounts of motion.

In the following table (Table 4.1.) we have averaged the bandwidth required over all X-minute intervals for each experiment. As was expected, the smaller averages are associated with the shorter intervals since the average bandwidth over the entire movie is about 0.36 Mbps.

Interval Size	**GOP Ave bw**	**Cost**
Every GOP	0.36 Mbps	2590 + 14512K
Every 7.5 secs	0.52 Mbps	3768 + 968K
Every 15 secs	0.58 Mbps	4202 + 484K
Every 30 secs	0.65 Mbps	4704 + 242K
Every 1 min	0.73 Mbps	5322 + 121K
Every 2 mins	0.82 Mbps	5970 + 61K
Every 5 mins	0.95 Mbps	6977 + 25K
Every 10 mins	1.05 Mbps	7931 + 13K
Base	1.80 Mbps	13080 + K

Table 4.1. – Costs for fixed interval approach

The costs in the above table are calculated using the cost function introduced in Section 3.5. where cost = Σ(no. of seconds * bandwidth) + (no. of renegotiations * K). We have summed the bandwidth used over each interval and added the cost of all renegotiations necessary. For example, if we renegotiate at the start of every GOP there will be 14512 renegotiations (equal to the number of GOPs). There are 13 intervals of 10 minutes in length hence there are 13 renegotiations in this case. In fact, the movie is just over 2 hours long, which implies that the 13^{th} interval is shorter than 10 minutes. In each case the bandwidth for the last interval has been calculated accordingly to accommodate the exact length of the movie. From the above, one can deduce that if the cost of renegotiation (K) is negligible, then making the intervals as small as possible, results in the lowest cost. In the following table (Table 4.2.), we have calculated the most economical intervals for the Starwars data according to various values of K. Note once again that K is measured in Megabits, implying that when K = 100 we assume the cost of renegotiation is equivalent to the cost of transporting 100 Mbits.

K - cost in Mbits	GOP
0	Every GOP (14512 negotiations)
5	Every 30 secs (242 renegotiations)
20	2 minutes (61 negotiations)
40	5 minutes (25 negotiations)
80	10 minutes (13 negotiations)
150	10 minutes (13 negotiations)
300	10 minutes (13 negotiations)
600	Base (1 negotiation)

Table 4.2. – The most economical intervals for various values of K

The most economical interval becomes wider as the value of K becomes larger. When the renegotiation cost (K) is large, one would want to keep the number of renegotiations to a minimum.

Using the GOP sizes as input data, if the renegotiation cost was negligible, and we were able to renegotiate the bandwidth for every GOP, we could achieve a saving of up to 80% over the maximum required bandwidth of 1.80 Mbps. Note that the cost of renegotiation would almost certainly be non-trivial and hence the 80% saving is highly unlikely. For example, with K = 1 there is no saving when we negotiate at every GOP.

These results tend to imply that network architectures should be designed to respond cheaply to fine-grained renegotiations, since it is always more economical to choose the smaller intervals when K is small.

4.3. Conclusions

By dividing the video stream into fixed intervals, we have been able to reduce the cost of transmitting the video over a network. We have run a number of experiments where various interval sizes have been used. Each interval in the experiments required a different bandwidth. At the start of each interval it was necessary to renegotiate a new bandwidth. Assuming a non-zero renegotiation cost, experiments with a large number of intervals had a higher transmission cost than those with fewer intervals. However, a large number of intervals generally implies that a smaller bandwidth is needed per interval. A compromise will then be necessary to optimise between a large number of intervals and a higher cost due to a large number of renegotiations.

Chapter 5

Variable Intervals

5.1. Introduction

Instead of fixing the interval length used for renegotiation, one could use different length intervals for each renegotiation of bandwidth. In this way, one could attempt to use the least amount of bandwidth for the longest period of time.

We have used variable intervals in two ways in this chapter. Firstly, we have introduced the use of heuristics, where one can shift each interval's boundaries closer together or wider apart in order to optimise the cost of the required bandwidth for that interval. Secondly, we have used scene changes in a movie to be the interval boundaries. We have assumed that scene boundaries are intuitively the best measure of change of content and hence, where a change in bandwidth would be necessary. If the scene changes were all known at the start of the movie, the cost and bandwidth portfolio for the movie could be known prior to sending the movie over the network. However, if this data is not known prior to sending the movie over the network, then the encoder or sender would need extra intelligence in order for it to detect scene changes and then calculate the required bandwidth and cost for each section of the movie.

5.2. Heuristics

Using the crude fixed-period negotiation as a starting point, we can devise heuristics to either extend or shorten some of the intervals, or to integrate adjacent intervals, or to break existing intervals into finer-grained sub-intervals. A heuristic is a technique that improves the efficiency of a search process, possibly by sacrificing claims of completeness.

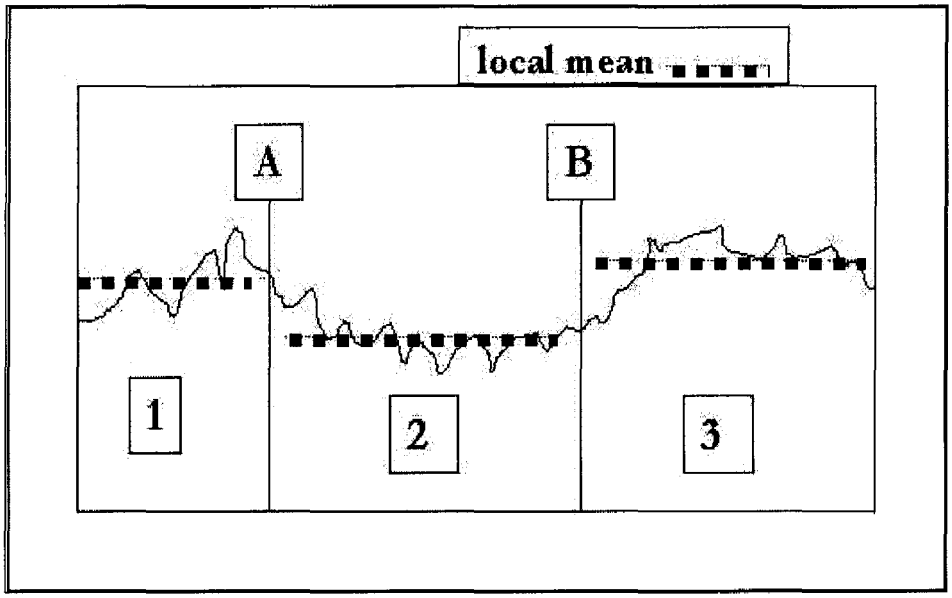

Figure 5.1. - A heuristic attempt at achieving optimum interval lengths

As Figure 5.1. implies, our heuristics should attempt to shift interval boundaries A and B to the left or right in order to determine the optimum sizes for the intervals 1, 2 and 3. In order to calculate the related cost we need to consider some set-up cost for each interval as well as the cost due to the required bandwidth and the length of each interval. In simple terms, we wish to use the least amount of bandwidth necessary for the longest period of time possible. Our costing algorithm introduced in Section 3.5.

would be appropriate. A heuristic algorithm will be used to determine these aforementioned optimum fragment sizes. Using such an algorithm, we are able to shift the interval boundaries in order to widen the interval requiring the least bandwidth.

The following are heuristic methods, mentioned in [Rich et al] that one could use to generate new solutions or search for other possible solutions to a given problem:

5.2.1. Generate-and-Test

This method generates a possible solution and then tests to see if it is valid. Any valid solutions are then tested to see if they are also optimal solutions. If a solution is not optimal then another solution is generated and the cycle continues.

5.2.2. Hill Climbing

This is a variant of the Generate-and-Test method. Feedback from the test procedure is used to help the generator decide which direction to move in the search space. Hill climbing is often used when a good heuristic function is available for evaluating states but when no other useful knowledge is available. Hill climbing is terminated when there is no reasonable alternative state to which to move. One would the check if the current state is a goal state. If it is not a goal state, some operator would be applied in

order to produce a new state. The method then checks to see if the new state is better than the current state. This method may fail to find a solution. It may terminate without reaching a goal state, by entering a state from which no better states can be generated.

5.2.3. Simulated Annealing

This method is a variation of the Hill Climbing method. It still seeks a goal state and will terminate if one is found. If the new generated state is not better than the current state, then it will become the current state with a given probability. If the new state is better than the current state then it will become the current state.

5.3. Boundary shifting approach

One possible approach would be to shift the boundaries of the intervals by a fixed amount and test to see if these 'new' intervals generate a lower cost than the original intervals. Using the fixed interval approach in Chapter 4, we have been able to generate different sized intervals, which produce lower costs. The algorithm we have designed generates interval sizes based on the 5-minute and 10-minute intervals used in Chapter 4. We were able to choose the amount by which we shift the interval boundaries in order to create 'new' intervals. If the new intervals produced a lower cost than the original interval sizes they have been kept otherwise they have been

discarded. The costs have been recorded and graphed in Figures 5.2. and 5.3.

The amounts by which the original boundaries are shifted are on the x-axes of the graphs, while the cost is on the y-axes. For example, using the original 10-minute intervals, we have chosen to shift the boundaries by 2 minutes in either direction and the cheapest cost produced was 7775. The first cost plotted in both graphs was the cost when the interval boundaries are not shifted at all. These costs correspond to the costs calculated in Table 4.1. of Chapter 4. Since the number of renegotiations do not change using this algorithm, no additional costs are involved.

In both graphs there is a distinct decrease in the cost to a minimum after which the cost begins to increase. This shows that there is a definite advantage in using this algorithm to reduce the cost of bandwidth utilisation. As long as the GOP sizes are known prior to sending the movie across the network then the network needs to do no extra calculations. All calculations can be done prior to transmitting the movie and a bandwidth profile can then be created.

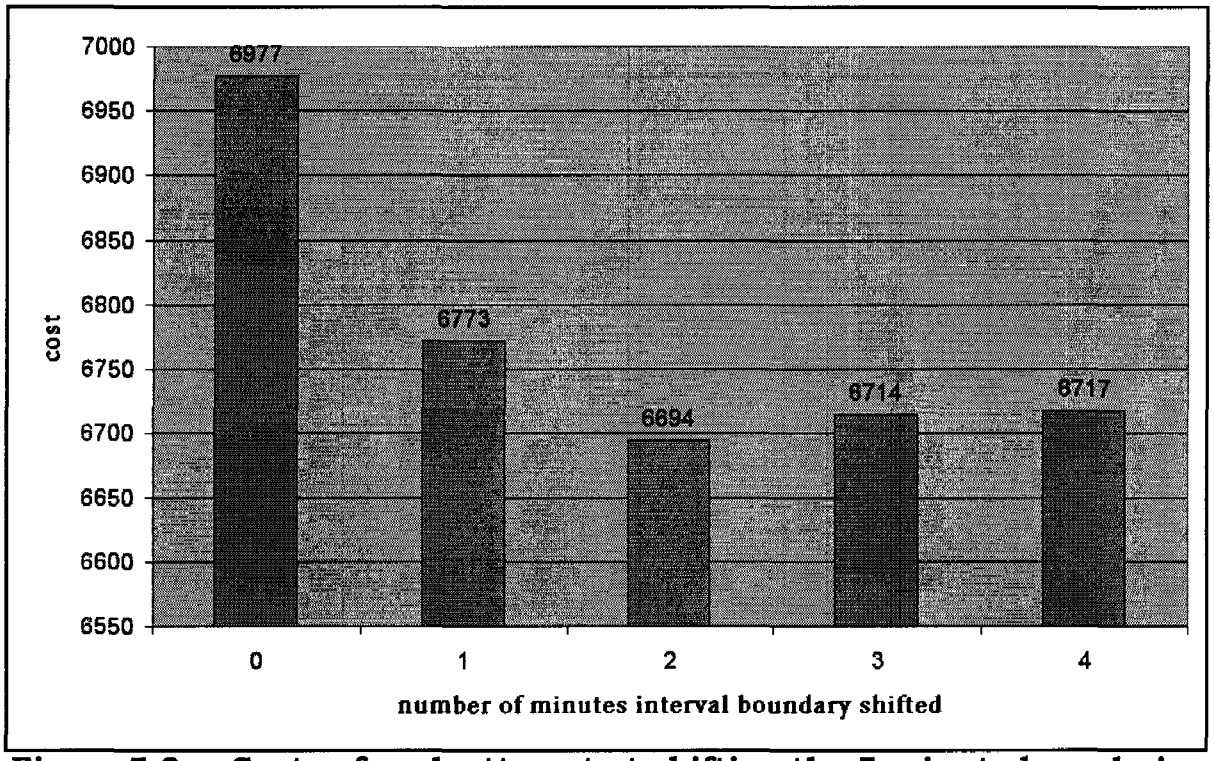

Figure 5.2. – Costs of each attempt at shifting the 5-minute boundaries by x minutes

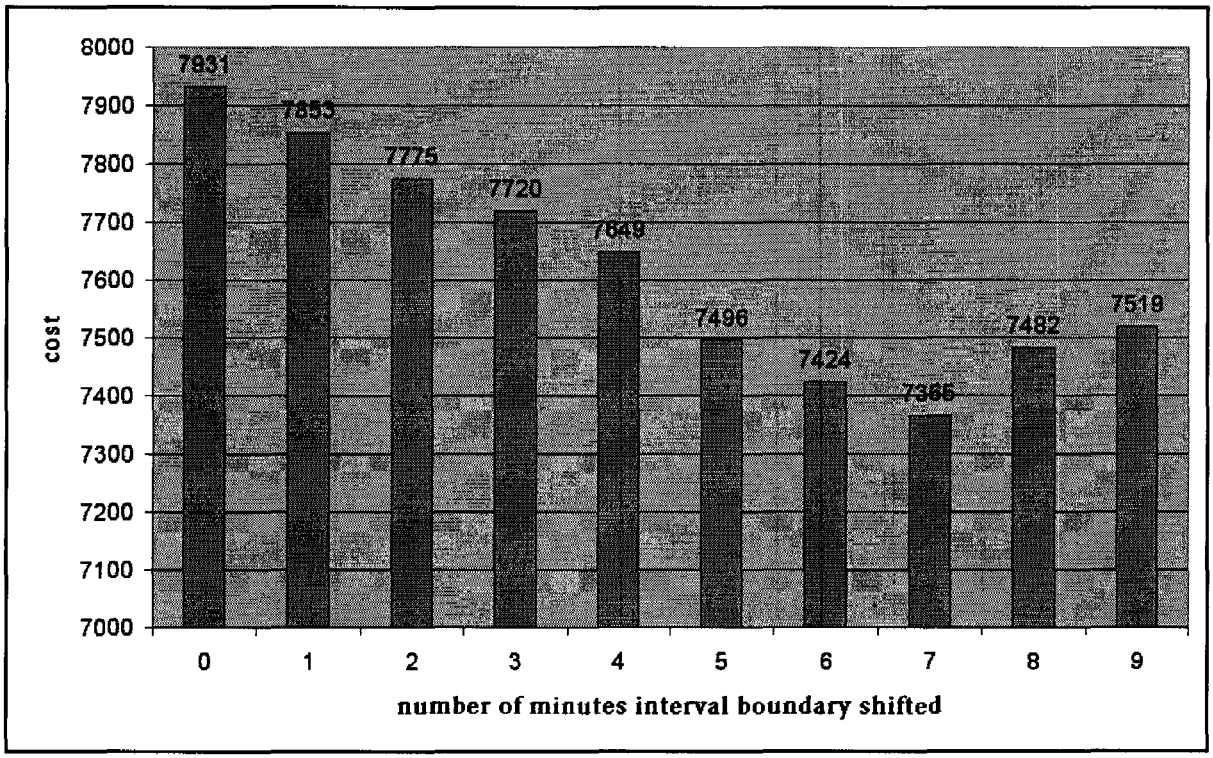

Figure 5.3. - Costs of each attempt at shifting the 10-minute boundaries by x minutes

Note that the costs in the above two graphs need to be increased by 25K and 13K respectively to achieve the total costs, which include the renegotiation cost.

In Figure 5.2. we see that the most economical approach requires shifting the original 5-minute interval boundaries by 2 minutes in either direction. This is equivalent to a 4% saving on the cost of the original interval sizes. The graph in Figure 5.3. shows that by shifting the 10-minute interval boundaries by 7 minutes in either direction we were able to reduce the cost to 93% of the original cost (a saving of 7%).

Heuristic algorithms like the one used above have potential to reduce the cost of the required bandwidth significantly enough to make the extra computation justifiable.

5.4. Quantisation approach

Another approach would be to use the quantisation method. Here we choose to map all GOP sizes to a number of pre-chosen sizes. In this way, we hope to reduce the number of bandwidth renegotiations necessary during the streaming of the video. This scenario is necessary if the network provider only allows renegotiation in fixed quanta, as is presently the case for ISDN services, in which new bandwidth can only be added in 64-Kilobit quanta.

Rather than use predefined quanta, we have created intervals based on the Starwars data. The minimum and maximum GOP sizes in the original Starwars data are 77500 bits and 945153 bits respectively. Within these borders we have created 5 intervals of equal size. With this knowledge we are then able to map the original GOP sizes to the maximum value of each interval. The following table (Table 5.1.) shows how the original GOP sizes are mapped to the chosen values.

Original values	Mapped values
77500 to 251030	251030
251031 to 424561	424561
424562 to 598091	598091
598092 to 771622	771622
771623 to 945153	945153

Table 5.1. – Mapping of GOP sizes (bits) to chosen values

All GOP sizes that fall into the interval given in the left-hand column are mapped to the value in the right-hand column. We now only have five possible values for the GOP sizes. This immediately reduces the number of possible renegotiations during the movie, unless every second GOP in the original data belongs to a different category to the GOP just before it. These 5 values would be able to carry the entire movie within the time allowed since the bandwidth required would always be less than or equal to the maximum value to which we are mapping.

By calculating costs for the above GOP sizes, we are able to achieve a cost of (3900 + 14473*K), where K is the cost of a single renegotiation. This cost will be less than the cost for renegotiating at every GOP, for all K > 33. If different values are chosen to which the original GOP sizes can be mapped, then different costs and hence different savings can be achieved.

5.5. Scene Changes

We could assume a bandwidth adjustment would be necessary when there is a scene change in a movie. We base this on the assumption of a change of content in a movie when there is a scene change. Scene changes are the best locations, intuitively, for bandwidth renegotiations. Complex scenes require more available bandwidth than scenes with little action. The content of a movie will therefore play a vital role in the cost of bandwidth usage and renegotiation. Using actual scene-change data, we have been able to calculate bandwidth requirements for a number of MPEG movies. This method assumes that we have access to actual content that allows us to detect the scene changes.

In some instances scene change data may not be directly available, if for example, the data were encrypted. In that case, one would have to resort to inspecting the size information in the stream. Unlike those algorithms that attempt to detect changes in the moving average, the scene change is more likely to be detected by looking at individual sizes of frames within a GOP to find data sizes that violate our expected norms for the pattern of I, B and P frames.

In general there are two kinds of scene changes. One is a camera break and the other is a gradual transition such as a fade or dissolve. There are a number of techniques available that can be used to detect a scene change.

Some of them extract features from each video frame, such as a colour histogram, in order to measure any content differences between consecutive frames. Comparing pixel differences between two frames can identify these changes. Comparisons based on histograms are also used. The colour histogram comparison and the chi-squared test of colour histograms are examples. Another approach is the pair-wise sub-frame colour histogram comparison where a video frame is divided into 4x4 rectangular regions. The comparison is then made on each pair of corresponding regions between two frames.

DCT coefficients can also be used in comparing two video frames. The number of motion vectors can be an indicator of data sizes that violate our norms. For each predicted frame (P and B frames), a number of motion vectors will be associated as explained in Chapter 1. The exact number associated with each frame depends on the residual error after motion compensation. If two frames are quite different, only a few motion vectors will be used for coding. Thus a scene cut can be detected based on the counts of motion vectors.

The detection of gradual transitions to new scenes is somewhat more complicated. The twin-comparison method requires the use of two cutoff threshold values. One would begin by comparing consecutive frames using a video segmentation method such as the colour histogram comparison. Once

a start frame for the transition period has been detected, frame differences are monitored until they become greater than the upper threshold. This is then taken to be the end of the entire transition period between scenes.

For the detection of a fade between scenes, one can use the fact that there is a linear variation in the luminance values while there is constancy in the chrominance values.

5.5.1. Costing of movies

Since a change of scene usually implies a change of content in a movie, we have decided to renegotiate the bandwidth required for a movie at the start of each new scene. MPEG videos with scene change data were found at http://www.cse.cuhk.edu.hk/~cflam/research/.

These were then used to calculate the costs given in Table 5.2. The other costs in the table were also calculated in order to compare how scene breaks compare with other methods of renegotiation. All costs were calculated using the original costing equation introduced in Section 3.5.

The scene change data was supplied in the following format:

1) filename of the movie
2) total number of frames
3) size of each frame i.e. 320 x 240 pixels
4) type and position of each scene change
   each scene change has three attributes: the type of scene change (a fade or cut etc), the frame number where the scene change began, the frame number where the scene change ended.

e.g.

```
1)    airwolf2.mpg
2)    412
3)    192 144
4)    [2,1,14,0,74,75,6,75,98,0,98,99,0,106,107,0,115,116,0,123,124,6,124,138,0,
       138,139,0,169,170,4,230,244,0,297,298,0,322,323,6,323,338,0,338,339,0,346,
       347,0,355,356,0,363,364,4,364,387,0,387,388]
```

The actual frame sizes were then supplied in another file along with the number of each frame. We were then able to match the start frame of a scene with the correct frame size and find the maximum frame size for every scene in the videostreams. These maximums were then used to calculate the required bandwidth for each scene. Using the above frame data we were also able to calculate the required bandwidth if a renegotiation was done after every frame as well as having a single negotiation of bandwidth at the beginning of the movie. No GOP information was available for these videostreams, hence we were unable to determine if there was any cost reduction due to the use of GOP sizes.

We have chosen just four of the available movies in order to analyse them completely.

In each case the number of scenes and number of frames per movie are given by the number of renegotiations necessary for each case. For example, in the Airwolf2.mpg video, there are 21 scenes and 412 frames.

Movie name	Per scene	Per frame	One renegotiation
Airwolf2.mpg	25.95 + 21 K	10.67 + 412 K	46.54 + K
Ad5.mpg	89.74 + 24 K	39.98 + 696 K	103.86 + K
Ad21.mpg	94.12 + 9 K	39.27 + 684 K	107.09 + K
Bobo.mpg	39.79 + 24 K	14.61 + 680 K	44.65 + K

Table 5.2. – Scene cost comparisons per movie

As would be expected the most economical method for renegotiating would be to do so at every frame, if the renegotiation cost was minimal. However for all four of the *.mpg movies, renegotiating at every scene would be more economical for all values of $K > 0.08$. Since these are such short movies, there is not much improvement from using a single bandwidth throughout the entire videostream. For example, renegotiating at every scene change in the ad21.mpg videostream is only economically viable for values of $K < 1.62$, and for the ad5.mpg video K needs to be less than 0.62. It is quite possible that renegotiation per scene would be more beneficial for full-length movies where the scene content varies a great deal.

5.6. Previous work

The papers mentioned below, have all introduced methods for detecting scene changes in MPEG compressed video streams. [YEUNG et al], in particular, mentions a number of different scene change detection methods including the Global Threshold Method, the Sliding Window Method and the Difference of frame difference Method. They also introduce many segmentation algorithms. The Pair-Wise Pixel Comparison Algorithm, the Difference of Colour Histogram Algorithm, the Motion Vectors Count Algorithm and the Simple Histogram Comparison are some examples.

Algorithms have been developed by [YEO et al] that detect abrupt and gradual scene changes. These algorithms use the DC sequence for detection, which can be extracted from the MPEG encoded video without having to decompress the stream in any way. Operating on the DC images offers significant computational savings. In order to gain even better detection, one could combine the results from the luminance DC and the two chrominance DC sequences. A scene change could be detected from frame i to i+1 if a change is detected on any of the 3 types of DC sequences, Y, Cb or Cr. [YEO et al] also introduced a more robust method for detecting a gradual transition between scenes. Here, they would use every frame and compare it to the following k^{th} frame. For example, compare frame i and frame i+k.

[NAGASAKA et al] experimented with various comparison techniques in order to detect scene changes. These functions were applied to only a portion of each image. Their method is robust against zooming and panning, but may fail to detect special effects like fading to a new scene.

Based on the hierarchical structure of frames and scenes, [KENDER et al] have derived a method for measuring probable scene boundaries. They have highlighted the underlying high-level organisational structures of videos, as presented in various textbooks on production, cinematography and film editing, which depend on many certain human expectations about the placement and timing of camera shots within a scene. There are thus limits, both maximum and minimum, on the number of viewpoints and on the lengths of camera shots that can be assimilated by an observer into a single scene.

[ARMAN et al] have developed a technique which can detect changes directly on the MPEG compressed data, without working on the original image sequences. [ZHANG et al] use the difference of histograms as well as global thresholds to detect scene changes.

[MENG et al] use the variance of DC coefficients in I and P frames and motion vectors information to characterise scene changes. [SETHI et al] use only the DC coefficients of I frames to perform hypothesis testing using the

luminance histogram. Unfortunately, the exact location of abrupt scene changes cannot be located with this method.

Using a variety of videos, [YEUNG et al] found that normalised colour histogram difference was a satisfactory measure of frame dissimilarity. This measure would be used to detect possible scene changes where bandwidth renegotiations could take place.

5.7. Conclusions

Since we can assume that dividing a video stream into fixed interval sizes would not generally result in the most economical interval lengths, we have attempted to use variable interval lengths in order to reduce the costs of frequent renegotiations of bandwidth. There are various methods available that enable one to divide a video stream into intervals of variable size.

Firstly, we have introduced a number of different heuristic approaches to our problem of how to divide a video stream into economical intervals. The idea behind the heuristic approach is to choose intervals that use the least amount of bandwidth for the longest period of time. We have attempted to reduce costs by shifting interval boundaries in either direction. New intervals were created if they resulted in lower costs than the original intervals. Using this method, our results show that there is a definite

advantage to shifting boundaries since in each of our experiments we were able to reduce the total cost of the movie.

Another heuristic method was to map all the GOP sizes onto a few chosen average values. Unfortunately, in order to choose these values, we would require the a priori knowledge of the original data. By only using a small number of possible GOP sizes, we immediately reduce the number of renegotiations required during the movie. Our experiments have shown that this method is very economical with the data we had at our disposal reducing the original costs for all K >= 3.

Our second method of creating variable length intervals was to use the lengths of scenes in a movie. Since we can assume that scene changes would imply a change in movie content within the movie, we have chosen these scene changes as appropriate locations for the renegotiation of bandwidth. For this method, one would require the scene change data; for example, the number of the frame at which a scene change begins. If the a priori data is not available then one would have to rely on algorithms that detect possible scene changes based on purely on the frame sizes of the movies. These algorithms would generally search for individual frames that violate the expected norms for the rest of the frames within a GOP. Our experiments have shown how renegotiating when a scene change occurs reduces the total costs substantially.

Chapter 6

Dynamic Renegotiation

6.1. Introduction

Dynamic renegotiation does not require the a priori knowledge of frame sizes that is demanded by the previous renegotiation methods. By looking a few frames ahead into the video stream we are able to adjust the bandwidth according to the moving average of the data. As the frame sizes increase, so the required bandwidth would increase. When this occurs, we are able to adjust the available bandwidth at run-time, i.e. any renegotiation takes place while the movie is being played. We have chosen to experiment with this method since no a priori knowledge is required and a renegotiation would only take place when one is required.

6.2. Moving Average

A simple algorithm, which monitors the moving average of the frame sizes or GOP sizes, can be used to trigger renegotiation. This method involves calculating a mean and a deviation (**s**) for the data. Since renegotiation is a local event (with respect to the MPEG data set), a global mean does not serve our research well. We have therefore chosen to modify the approach by using a local mean (as a moving average). We have used (mean + 5*s) as the maximum allowable bandwidth, while (mean ± 3*s) is used as the upper

and lower boundaries respectively. These values have been chosen according to Tchebysheff's Theorem (see [MENDEN et al]), which gives a lower bound to the probability that a variable (frame size or GOP size in this case) falls in an interval of the form ($\mu \pm m\sigma$), where μ is the mean and σ is the standard deviation. The lower bound is then given by $(1 - 1/m^2)$. Therefore the probability that a GOP size falls within the maximum allowable bandwidth would then be $(1 - 1/25) = 0.96$.

In general, if we can predict a buffer underflow we could then increase the bandwidth (renegotiate) and similarly decrease the bandwidth when a buffer overflow is predicted. If the moving average increases and crosses the upper bound (mean + 3*s), a new mean and a new standard deviation are calculated and a renegotiation takes place. Using these new values, a new maximum bandwidth is calculated. This would also occur if the moving average decreased below the lower bound (mean–3*s).

If only a small amount of buffering was available, then one would have to calculate the moving average on a small amount of data, for example every 6 frames (half a GOP) which equates to a quarter of a second of our movie time. However, adequate buffering would allow one to use GOP data. With n=4, this would equate to 2 seconds of our movie time, since one GOP is equivalent to a half a second of our movie data (Starwars). Our data set of

moving averages does show long periods where no renegotiations are necessary.

In the following table (Table 6.1.), the bandwidth value is calculated using (mean + 5*s), while the upper and lower bounds are calculated with (mean ± 3*s). The bandwidth values are the moving averages for the GOP data (n=4). All the values are measured in bits.

Lower bound	Mean	Upper bound	Bandwidth
0.020	0.216	0.514	0.678
0.020	0.334	0.514	0.678
0.020	0.432	0.514	0.678
0.020	0.503	0.514	0.678
0.190	0.554	0.918	1.161
0.190	0.475	0.918	1.161
0.190	0.409	0.918	1.161
0.190	0.362	0.918	1.161

Table 6.1. - moving average and related boundaries (GOP data)

We are now able to calculate a cost according to the original cost function in terms of K, the renegotiation cost.

Cost = (no. of renegotiations * renegotiation cost(K)) + Σ(no. of seconds * bandwidth)

For the GOP data, the cost of carrying the entire movie and renegotiating dynamically when required, would be 17135 + 7K for all K >= 0.

In order to compare the costs involved when using the fixed interval approach and the dynamic renegotiation approach for the GOP data, we have constructed the following table showing the relevant costs for different values of K.

K	Every 15 seconds	Every 2 minutes	Every 10 minutes	Dynamically
0	4202	5970	7931	17135
20	13882	7190	8191	17275
40	23562	8410	8451	17415
80	42922	10850	8971	17695
150	76802	15120	9881	18185
300	149402	24270	11831	19235
600	294602	42570	15731	21335

Table 6.2. – Cost comparison for fixed intervals vs dynamic renegotiation for various K

For very large values of K, the dynamic approach would be one of the better approaches for our data set, when compared to the other fixed interval

approached we have used. Due to the large area between the upper and lower bounds, there are a very few number of renegotiations for the Starwars data. By adjusting these bounds one would be able to alter the number of renegotiations during a movie and thereby adjusting the cost as well.

6.3. Conclusions

Our approach in Chapter 6 does not use the a priori knowledge of frame sizes. We are able to dynamically adjust the required bandwidth at run-time using the moving average of our Starwars data. If the moving average exceeds some upper bound (based on the mean + 3 * standard deviation (s)) the required bandwidth is increased. We have set upper and lower boundaries at (mean ± 3*s) while the maximum required bandwidth is set at (mean + 5*s).

If only a small amount of buffering was available we would have to calculate the moving average on the frame data using about 6 frames, otherwise, with more buffering one could use say 4 GOPs in order to calculate the moving average.

Conclusion

In our investigation of techniques that can be used to bypass some of the difficulties associated with variable bit rate traffic, we have focussed our attention on video-on-demand servers. We have not attempted to find statistical models for MPEG video data.

We have shown that short-term as well as long-term variations occur in MPEG videos. The existence of these variations is not uncommon in MPEG encoded videos as can be seen by much previous research done on this topic.

The above-mentioned variations result from the variable bit rate traffic which is inherit in MPEG encoded video. The short-term variation is due to the difference in the frame sizes of the movies. Because of the way MPEG is encoded (with I, B and P frames), we can expect a video stream of MPEG encoded data to vary substantially in its short-term bandwidth requirements.

In order to alleviate this problem, we have introduced a small amount of buffering at the receiver end of the network. This buffer can smooth out much of the short-term variation evident in a videostream. By grouping frames into naturally occurring Groups of Pictures (GOP's) and buffering an

entire GOP before displaying the frames, we have been able to reduce the required bandwidth for our MPEG video under investigation significantly. With the above result in mind, we have also focussed much attention on the long-term variation that occurs in most video streams. We attempt to accommodate this variation by changing the bandwidth over time.

This type of variation is due to the change in content within a movie. Scenes in a movie that contain large amount of motion and complexity tend to result in above average frame sizes. As scenes move from high complexity to low complexity, the frame sizes also become smaller in size.

In order to alleviate the problems associated with this type of variation, we have introduced a number of renegotiation techniques, which are able to reduce the amount of required bandwidth over the entire videostream. We have proposed a costing algorithm in order to evaluate the different techniques used. Most of these techniques have used the a priori knowledge of the frame sizes.

The proposed techniques include dividing the video stream into fixed intervals and calculating the bandwidth required per interval. We have also used a variation on this method by dividing the videostream into intervals of different lengths based on heuristic methods as well as on scene changes.

Finally, we have shown how to cope with long-term variation without having to use the a priori knowledge of frame sizes. By using a moving average and setting upper and lower bandwidth limits, we have been able to propose a system of renegotiating bandwidth, which is able to carry the entire movie at a reduced cost.

References

[ARMAN et al] F.Arman, A.Hsu and M.Y. Chiu. "*Feature management for large video databases*". Storage and Retrieval for Image and Video Databases, 1993, volume SPIE 1908.

[C-CUBE] C-Cube Microsystems: MPEG Technical Overview. http://www.c-cube.com/technology/mpeg.html

[COLLINS] Collins Dictionary of the English Language, 2nd Edition. Collins London and Glasgow 1986.

[DAN et al] A.Dan, D.M.Dias, R.Mukherjee, D.Sitaram and R.Tewari. "*Buffering and caching in large-scale video servers*". IBM Research Division.

[DOULAMIS et al] N.D.Doulamis, A.D.Doulamis, G.E.Konstantoulakis and G.I.Stassinopoulos. "*Performance models for multiplexed VBR MPEG video sources*". National Technical University of Athens.

[FENG et al] W.Feng and J.Rexford. "*A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video*". Ohio State University and AT&T Labs Research.

[GARRETT et al] M.W.Garrett and W.Willinger. "*Analysis, Modeling and Generation of Self-Similar VBR Video Traffic*". (Sept 1994). Proceedings of the ACM Sigcomm '94, pp. 269-280.

[JAHANIAN et al] W.Feng, F.Jahanian and S.Sechrest. "*An optimal bandwidth allocation strategy for the delivery of compressed prerecorded video*".

[KENDER et al] J.R.Kender and B-L.Yeo. "*Video scene segmentation via continuous video coherence*". Department of Computer Science, Columbia University and IBM T.J. Watson Research Center.

[KRUNZ et al] M.Krunz and S.Tripathi. "*Exploiting the temporal structure of MPEG video for the reduction of bandwidth requirements*". Department of Computer Science, University of Maryland.

[LAM et al] S.S.Lam, S.Chow and D.K.Y.Yau. "*An algorithm for lossless smoothing of MPEG video*". Department of Computer Science, University of Texas at Austin.

[LI et al] S.Sengodan and V.O.K.Li. "*A generalised grouping and retrieval scheme for stored MPEG video*". Communication Sciences Institute, University of Southern California Los Angeles.

[MENDEN et al] W.Mendenhall, D.D.Wackerly, R.L.Scheaffer. "*Mathematical Statistics with Applications Fourth Edition.*" PWS-Kent Publishing Company 1990.

[MENG et al] J.Meng, Y.Juan and S.F.Chang. "*Scene change detection in a MPEG compressed video sequence.*" Digital Video Compression: Algorithms and Technologies, volume SPIE 2419, February 1995.

[NAGASAKA et al] A.Nagasaka and Y.Tanaka. "*Automatic video indexing and full-motion search for object appearances*". Proceedings of the IFIP TC2/WG2.6 Second Working Conference on Visual Database Systems, September 30-October 3 1991.

[NORIAKI et al] Noriaki Kamiyama and Victor O.K. Li. "*Renegotiated CBR transmission in Interactive Video-on-Demand System*". Communication Sciences Institue, University of Southern California.

[RICH et al] Elaine Rich and Kevin Knight. "*Artificial Intelligence 2nd Edition*". McGraw-Hill, 1991.

[ROSE] O.Rose. "*Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems*". (February 1995). Report no 101, Institute of Computer Science, University of Würzburg.

[SETHI et al] I.K.Sethi and N.Patel. "*A statistical approach to scene change detection*". Storage and Retrieval for Image and Video Databases III, volume SPIE 2420, February 1995.

[YEO et al] Boon-Lock Yeo and Bede Liu. "*A unified approach to temporal segmentation of motion JPEG and MPEG compressed video*". Proceedings of the Second International Conference on Multimedia Computing and Systems, May 1995.

[YEUNG et al] M. Yeung and B. Liu. "*Efficient matching and clustering of video shots*". Proceedings of the International Conference on Image Processing, Volume I, pages 338-341, 1995.

[ZHANG et al] H.J.Zhang, A.Kankanhalli and S.W.Smoliar. "*Automatic partitioning of full-motion video*". Multimedia Systems, volume 1, July 1993.

Below are some sites where MPEG movies can be obtained:
ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/traces/
ftp://ftp.microsoft.com/Products/NetShowTheater/MPEG1/
ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/mpeg/movies/