LOSSLESS VIDEO MULTIPLEXING FOR TRANSPORT OVER COMMUNICATION NETWORKS

By

Chan Hang Fung

A Thesis

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Information Engineering

The Chinese University of Hong Kong

June 1997

Acknowledgement

I have to thank Cherry Fu and my family for their understanding, support and encouragement.

I am very grateful to Prof. Soung Liew for his supervision in these two years. His advice and comments are invaluable to my studies.

I would also like to thank the members of Broadband Communication Laboratory. They are very friendly and helpful. Without their support, I would not have completed this thesis.

Abstract

This thesis proposes and investigates a *Lossless Video Aggregations System* (LVAS) for multiplexing stored variable-bit-rate videos for transmission over a shared communication channel. While there have been previous work on *lossless adaptation* and smoothing of a single video stream for transmission, this is the first study on how multiple videos can be jointly smoothed and adapted to achieve higher efficiency in bandwidth usage. We show that compared with the straightforward approach of transmitting individual videos independently over separate constant-bit-rate channels, the proposed method requires significantly less elastic buffer at the receiver ends. In addition, compared with naive multiplexing methods that do not take into consideration of the relative and varying characteristics among the video streams, our method can also achieve much higher multiplexing gain. This thesis goes into many detailed issues related to *lossless video aggregation.* Among the subjects studied are a number of alternative bit-rate allocation methods and a detailed comparison among them, efficient implementation of aggregation in practice to achieve high-speed computation, implications of lossless aggregation for call admission, and modification of the aggregation process for interactive control in video-on-demand applications.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview of video transmission

Video information is often compressed for efficient storage and transmission. Many video compression methods, including the popular MPEG (Moving Picture Expert Group)[3] encoding, produce video that is intrinsically variable-bit-rate (VBR) because of: (1) the difference in information contents of different frame; (2) the use of *intraframe* and *interframe* coding for successive video frames. *Intraframe* coding compresses a frame according to the information within the frame without reference to others, while *interframe* coding compresses a frame with reference to adjacent frames to reduce the bits needed. Therefore, interframe-coded frames would typically have fewer numbers of bits than intraframe-coded frames. As coding switches between the two modes, the video source outputs a varying number of bits.

In the international video coding standard MPEG, the I-frames are *intraframes*-coded and the P- and the B-frames are interframe-coded. The largest

frame can be tens of times larger than the smallest one.

Video information can be transmitted over a communication network using either a VBR or a constant-bit-rate (CBR) channel. CBR transmission has many advantages from the networking standpoint: multiplexing, bandwidth allocation, user/network contractual agreement, and network-usage tariff are all simpler under the CBR transmission framework. A most interesting issue is therefore how to adapt VBR video for transmission over a CBR channel to leverage the high-compression efficiency of VBR compression and operational simplicity of CBR transmission.

A common CBR-transmission strategy that has been extensively studied [1, 2, 4, 5, 7, 8] is to adaptively compress and code the video so that the data stream produced is CBR. When the scene becomes very active or complex, and successive video frames threaten to produce more data than that can be accommodated by the CBR channel, the image quality will be reduced to bring the output data rate in line with the CBR channel rate. Because of the possible loss of image quality, this is sometimes referred to as *lossy adaptation.* A typical scheme is to employ a buffer to smooth the encoding process. The video frames from the encoder are fed into the buffer at a variable rate in accordance with the bits contained in them, and the data in the buffer are taken out at a constant rate for transmission on a CBR channel. The buffer-occupancy level is used as a feedback to the encoder to adjust the target image quality of the future frames to prevent buffer overflow or underflow. A major drawback of *lossy adaptation* is its susceptibility to fluctuations in video quality which may be visually disturbing.

In contrast to *lossy adaptation*, there is another approach called *lossless adaptation* which transmits the original video without sacrificing quality. The

video is encoded at a constant quality beforehand, and the VBR traffic is then fed into a CBR channel. Because the video frames are of different size and a CBR channel is used for transmission, the interval between the arrivals of adjacent frames at the receiver is not a constant. To absorb this variability, a large elastic buffer is placed before the decoder to intentionally introduce a delay between the arrival and display of a frame. To avoid underflow (which corresponds to late arrival of frames to be displayed) and overflow (which causes loss in video information) of the buffer, the video source must have *a priori* knowledge of the bit-rate profile of the video as well as the receiver buffer size so that it can schedule the transmission properly. For this reason, *lossless adaptation* is generally targeted for applications in which the video has been pre-encoded and stored, such as PPV (Pay-Per-View) or VOD(Video-On-Demand) services. The bit rate information can be collected in an off-line manner once the video has been captured and compressed. In the MPEG standard, for instance, the frame header identifies the boundary of each frame. The number of bits in each frame can be easily obtained by a one-pass scanning of the compressed video.

In the context of the classification as outlined above, the focus of this thesis is on *lossless adaptation.* In particular, we are interested in exploring how several VBR video streams should be jointly adapted to improve system performance in terms of end-to-end delay and receiver buffer size.

1.2 Previous work on lossless video transmission

A conventional method for lossless transmission of video is to reserve a CBR channel bandwidth equivalent to the peak rate of the VBR video. This method suffers from inefficiency in bandwidth usage, especially when the bit rate is highly varying and there is a huge difference between the peak rate and the average rate. Assuming the channel delay is a constant (which is easy for the network to guarantee for CBR channels), this method ensures that video frames arrive at the receiver in a *constant-frame-rate* manner, that is, the interval between the arrivals of successive frame is a constant. Thus there is no need for a large receiver buffer to smooth out the frame arrivals.

References [9, 11] proposed a method that is more bandwidth efficient. The main concept is to pre-send a large amount (but not all) of video data to the receiver before the actual display time. A CBR channel is used to deliver the rest of the data at a constant rate in such a way that the data will arrive at the receiver before its display time. In particular, the CBR channel bandwidth is fully utilized to transmit video data at all time. A large receiver buffer is used to smooth out the video frames which arrive in a *variable-frame-rate* manner for display in a *constant-frame-rate* manner. A major concern with this approach, however, is that the receiver buffer must be very large (e.g., several tens of megabytes)[9].

On a philosophical level, this method employs *temporal sharing* of bandwidth among successive frames to smooth out the rate variation. There are many situations in which many video streams are to be transmitted over a network channel

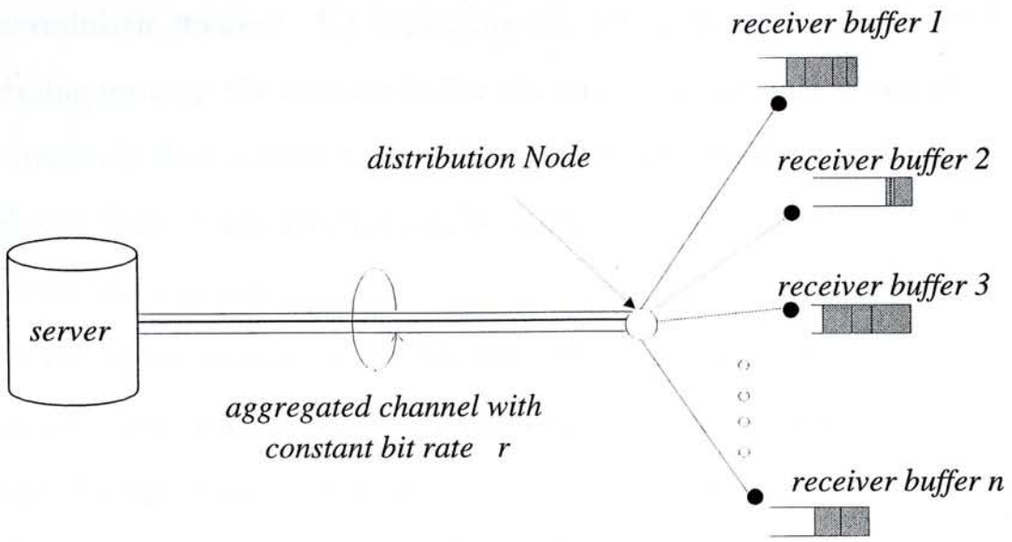

Figure 1.1: Architecture of the Lossless Video Aggregation System

simultaneously. In these situations, *spatial sharing* of bandwidth among several streams can be exploited to achieve further performance gain. The previous work has not considered this aspect, and this forms the major thrust of this thesis. Specifically, we propose a *Lossless Video Aggregation System*(LVAS) which jointly adapts a number of video streams. As shall be shown, using LVAS, the receiver buffer requirement can be reduced significantly.

1.3 Central theme of thesis — Lossless video Aggregation

Figure 1.1 shows a setup for the *lossless video aggregation system.* It aggregates a number of stored video streams together for transmission to a number of client receivers through a common transmission channel. The server makes use of the bit rate information of the video streams to schedule their delivery in

a deterministic manner. By exploiting the bit-rate profiles intelligently in the scheduling process, the receiver buffer size can be significantly reduced compared with methods that transmit the video streams independently. A benefit is that the elastic delay at the receiver can be reduced to improve interactivity between the client and the server.

As the video sequences are pre-encoded and stored in the server, there is no need to have a transmitter buffer. However, receiver buffers are needed to reshape the traffic back into the original VBR videos so that the display frame rate is a constant. Since the bit-rate profiles of the requested videos are known, the server can calculate the buffer occupancies of the clients at all time. It can schedule the transmission so that the underflow probability as well as the buffer requirement is minimized.

Application example of lossless video multiplexing in communication networks

There are many video-delivery systems in which LVAS can be applied. In particular, it is not necessary for the individual receivers to be located at the same place. This is illustrated with the three examples in Fig. 1.2

ATM Network

Fig. 1.2(a) shows a video-on-demand system in which a server transmits video streams to a number of receivers at different homes. Each stream is targeted for one of the receivers. The video streams are first transmitted using *lossless aggregation* via a CBR channel to a distribution node, whereupon the video streams are separated and delivered to the targeted homes individually.

In a public network, the distribution node is a *remote node* to which the

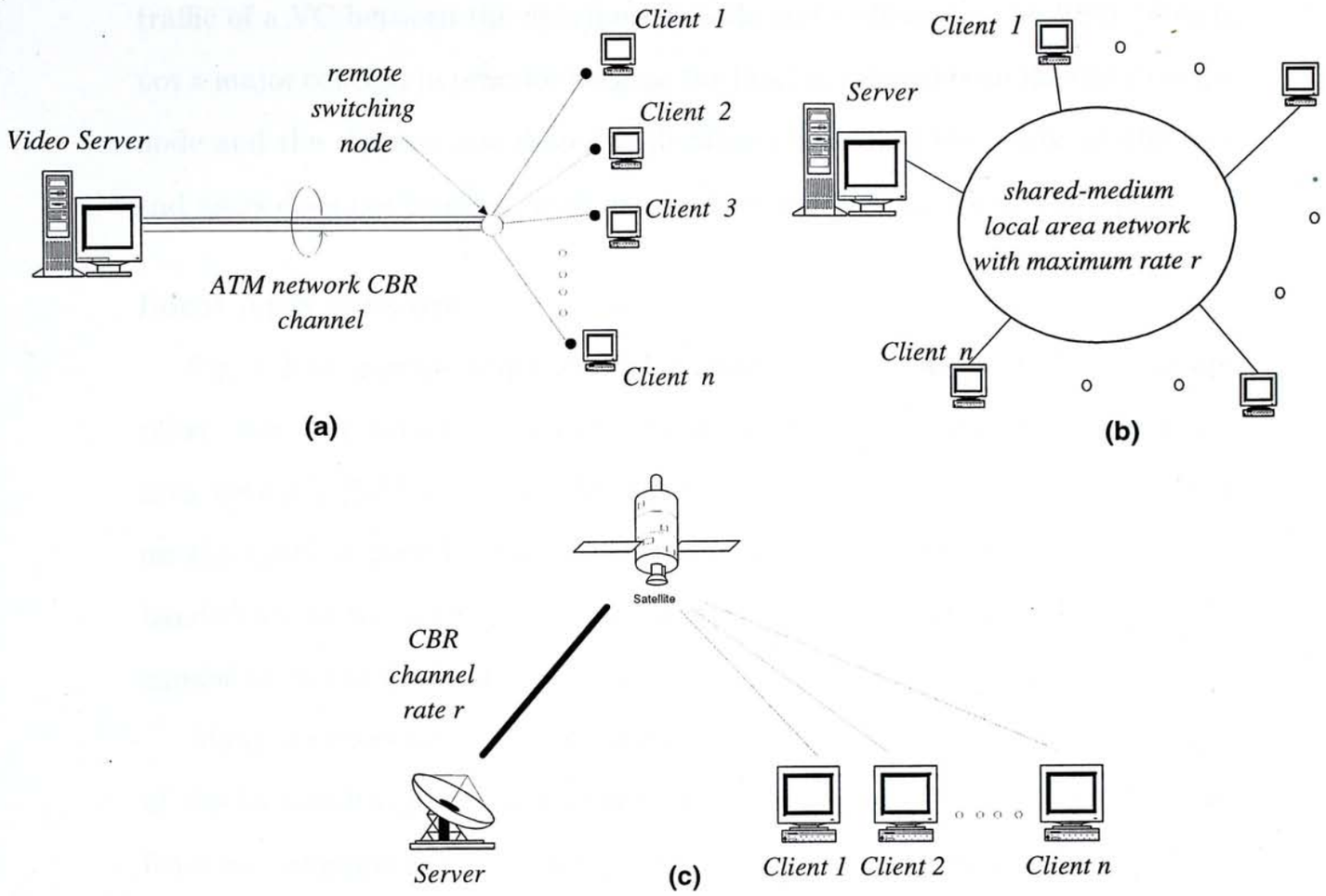

Figure 1.2: Application examples of the Lossless Video Aggregation System

subscribers in a neighborhood are connected. The video vendor may be located either in or away from a central office, and it may be serving an area covered by several distribution nodes. Video streams targeted for the same distribution node (but different subscribers) may be subjected to the aggregation scheme.

Suppose that the video-on-demand system is deployed over a public ATM network. Then, the CBR channel could be a virtual path (VP) that the video vendor leases from the network provider, and the individual video streams could be carried on different virtual channels (VC) within the VP. Note that the data

traffic of a VC between the distribution node and a client may be VBR. This is not a major concern in practice because the link bit rate between the distribution node and the client is not shared with other clients and the traffic of different end users does not interfere with one another beyond the distribution point.

Local Area Network

Fig. 1.2(b) shows another network scenario in which aggregation can be applied. Here, the server is connected to the clients over a shared-medium local-area network (LAN). If only the server is transmitting, the total bandwidth on the LAN is used by the server. Alternatively, a certain amount of fixed bandwidth on the LAN can be allocated for the server, if the network has the capability of bandwidth reservation.

Many standard LAN protocols nowadays can provide a lower bound estimates of the bandwidth, provided that only a server is transmitting while others are listening. Examples are dedicated ethernet, token ring, FDDI, etc. Furthermore, some new LAN standards can even provide guaranteed bandwidth service (e.g., Isonet and 100VG-AnyLAN). *Lossless aggregation* can, therefore, be applied in these types of networks.

One can also envision a hierarchical video delivery system with a wide-area network (WAN) and LANs. The server is connected to the WAN, and it sends video streams to gateways that are connected to the WAN as well as LANs. Through the gateways, the video streams are distributed to individual clients over the LANs.

Satellite Transmission

Another scenario is that of satellite transmission (fig. 1.2(c)). A video server may send video streams over a CBR up-link to a group of clients at different locations. Since the satellite bandwidth is a precious resource, *lossless aggregation* greatly reduces the up-link bandwidth given a certain amount of receiver buffer, and therefore LVAS can be considered as an efficient method for transmission of videos in satellite networks.

As a reference model, this thesis will focus on the picture in Fig. 1.2(a), where a server transmits a bunch of videos to clients through a shared link. However, there is no reason why this concept cannot be extended to the case where the shared channel traverses a number of nodes and links, with each containing traffic from other sources. All it needs is that a CBR logical channel be reserved through the path for our system to operate.

Another point that is worth noting is that *lossless aggregation* only makes use of the bit-rate profiles of the video for its operation. In particular, no assumption is made on the formats of the video streams. This system can therefore be applied to any video formats, including the case where there is a mix of different formats among the video streams to be adapted.

1.4 Organization of thesis

The thesis is organized into several chapters. Chapter 2 introduces the framework for the discussion of *Lossless Video Aggregation System.* Chapter 3 presents the buffer minimization problem using LVAS. We will verify the effectiveness of LVAS for the reduction of receiver buffer size. Chapter 4 discusses a number

of different bit-rate allocation methods that can be used in LVAS in practical settings. Experiments are performed to compare the performance of the different methods. Chapter 5 addresses the issue of call admission and interactive control. Some modifications of the LVAS algorithm to accept users' interactive commands are proposed. Conclusions and a number of interesting topics for future research are given at the end of the thesis.

Chapter 2

Conceptual framework of Lossless Video Aggregation System

2.1 Review: Transporting single VBR stream using a CBR channel

Before presenting the framework of aggregation, we will first review the key ideas of transporting single VBR stream i over a CBR channel in [9]. With reference to Fig. 2.1, the transmitter sends a video bit stream to the receiver, and the arrival time of the first bit at the receiver is $t = 0$. To build up some data in the receiver's elastic buffer, a delay is introduced and the display of the video does not start until $t = d_i$.

The shape of the display curve $A_i(t, d_i)$ is dictated by the intrinsic bit-rate

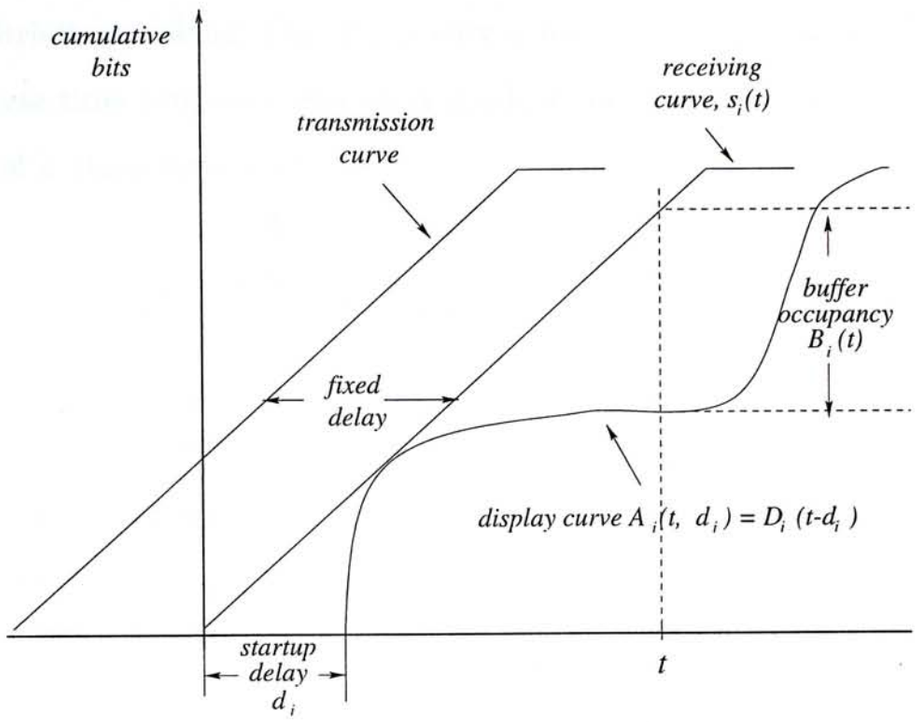

Figure 2.1: Illustration of the transmission of a video stream over a CBR channel

characteristics of the video and is independent of the design of the delivery system. The receiving curve $s_i(t)$, however, depends on how the video is being transmitted at the source as well as the delays introduced by the network. In this thesis, we assume that the delay jitter of the CBR channel is negligible so that if the video is transmitted at a constant rate, it is then received at a constant rate.[1] Between arrival and display, the video data is stored in a buffer at the receiver. As shown in Fig. 2.1, the difference between the arrival and display curves in the vertical direction is the amount of data in the buffer at time t. Since the arrival rate for the receiver buffer is constant and the departure rate is variable, the buffer occupancy will fluctuate over time.

[1] If the network delay is not negligible but is constant, the arrival curve is right-shifted. If the network delay is not constant but can be bounded, an additional amount of receiver buffer is required to smooth out the delay jitter. In general, the discussion here is still valid, albeit a small amount of modification is needed.

Strictly speaking, Fig. 2.1 is only a macroscopic picture of the underlying discrete-time processes described mathematically as follows. The cumulative bits of a video stream i is

$$D_i(t) \triangleq \begin{cases} 0 & : \quad t < 0 \\ \sum_{j=0}^{t} X_i(j) & : \quad 0 \leq t < N_i - 1 \\ \sum_{j=0}^{N_i-1} X_i(j) & : \quad N_i - 1 \leq t \end{cases} \tag{2.1}$$

where $X_i(j)$ = size of the j-th frame, starting at 0, in sequence i and N_i = number of frames in sequence i. The display curve with startup delay d_i is then only a constant right shift of $D_i(t)$,

$$A_i(t, d_i) \triangleq D_i(t - d_i). \tag{2.2}$$

For a CBR channel with rate r, the receiving curve is given by

$$s_i(t) = \begin{cases} rt & : \quad t < t' \\ rt' & : \quad t \geq t' \end{cases} \tag{2.3}$$

where $t' = \lceil \frac{A_i(N_i+d_i-1,d_i)}{r} \rceil$ is the time by which all data in the video has been received. Here we make the assumption that the bits for each time slot are arrived and consumed at the beginning of this particular time slot.

To prevent buffer underflow (i.e., the situation in which the data to be displayed has not yet arrived), the receiving curve must be above the display curve at all time. For a given fixed d_i, this means that the receiving rate r cannot be too low. Specifically, the buffer occupancy of sequence i with startup delay d_i at time t is

$$B_i(t) \triangleq s_i(t) - A_i(t, d_i) \geq 0 \quad \forall t \tag{2.4}$$

The maximum buffer occupancy over time, $\max_t B_i(t)$, dictates the buffer size required at the receiver. It is desirable to make r as small as possible while

satisfying the above relationship because the required receiver buffer size can then be minimized. For the smallest possible r, there is always a point at which the receiving and display curves almost touch each other; if not, r can be reduced further. More precisely, the optimal r is given by

$$r = \max_{0 \le t < d_i + N_i} \left\lceil \frac{A_i(t, d_i)}{t} \right\rceil \tag{2.5}$$

Given the above optimal choice of r, the next question is how $\max_t B_i(t)$ changes with d_i. From the macroscopic picture in Fig. 2.1, the reader can check intuitively that there is an optimal d_i at which $\max_t B_i(t)$ is minimized. In [9], experiments have been conducted over the movie *Star War*. It was found that the buffer needed is about 22.3Mbyte, or 6% of the total video size, with a build up delay of 37 seconds. The large buffer size is a concern in practical implementation, and the straightforward CBR transmission will need to be modified. *Lossless Video Aggregation System* is designed to address this problem.

2.2 Lossless aggregation of VBR streams

It turns out that the receiver buffer requirement can be reduced when a number of videos are transmitted together using a CBR channel, as shown in Fig. 2.2. Each video may have a time-varying receiving rate. However, to fully make use of the channel rate, the sum of the receiving rates of all videos must equal the CBR channel rate. To prevent underflow at all receivers, it is necessary for the aggregate display curve to be below the aggregate receiving curve. We can write the aggregate display curve of the n video streams as

$$A(t, \vec{d}) = \sum_{i=1}^{n} A_i(t, d_i), \tag{2.6}$$

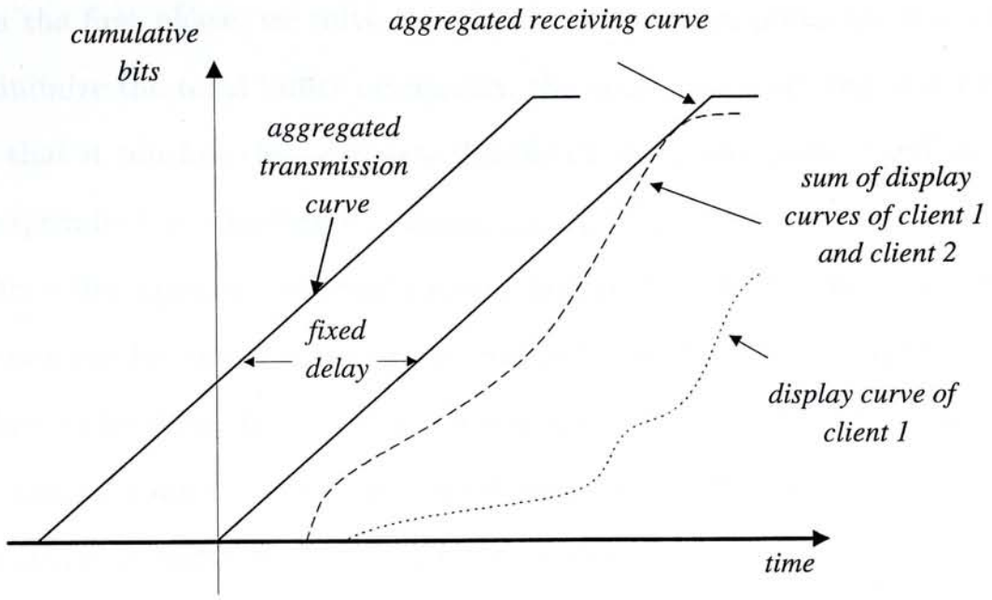

Figure 2.2: Illustration of two videos multiplexing together into a CBR channel

where $\vec{d} \triangleq \{d_1, d_2, \ldots, d_n\}$ is the vector of startup delays at the receivers. Without loss of generality, we assume $N_1 + d_1 \leq N_2 + d_2 \leq \ldots \leq N_n + d_n$.[2] The aggregate receiving curve for a CBR with rate r is

$$S(t) = \begin{cases} rt & : \quad t < t'_n \\ rt'_n & : \quad t \geq t'_n \end{cases} \tag{2.7}$$

where $t'_n = \lceil \frac{\vec{A}(N_n + d_n - 1, \vec{d})}{r} \rceil$ is the time by which all video data have been received at all receivers. The sum total of buffer occupancies at all receivers is then $B(t, \vec{d}) \triangleq S(t) - A(t, \vec{d})$.

In addition to the above global consideration, we need to devise a way to apportion the aggregate rate r to each of the videos i such that its receiving curve is above its corresponding display curve. There are many ways to do this, and the following presents a two-phase approach.

[2] In other words, the cardinality of the sequences is determined by this relationship.

In the first phase, we solve the global transmission problem. For a given $\vec{d}$, to minimize the total buffer occupancy, the aggregate receiving rate can be set such that it touches the aggregate display curve at one point. Similar to (2.5), it is equivalent to choosing $r = \max_{0 \le t < d_n + N_n} \lceil \frac{A(t,\vec{d})}{t} \rceil$.

Once the aggregate receiving rate r is determined, the buffer occupancy at any time can be shared among those receivers that have not completely received all their video data. In the second phase, our problem is to determine $B_i(t) \forall i, t$ such that the worst-case buffer occupancy at all time and for all sequences, $max_{i,t} B_i(t)$, is minimized subject to two constraints:

- Constraint 1: $\sum_i B_i(t) = B(t, \vec{d})$ and $B_i(t) \geq 0 \quad \forall t$;
- Constraint 2: The individual receiving curve $s_i(t) = A_i(t, d_i) + B_i(t)$ is non-decreasing $\forall i, t$.

The first constraint is to ensure that no bandwidth is wasted and the aggregated transmission rate is fixed to be r while no underflow would occur. The second constraint is to ensure that the curve can be realized in practice. As s_i is the *accumulated* receiving curve for sequence i, it cannot be decreasing in any case. Once $B_i(t)$ has been determined, $s_i(t)$ can be constructed and the server can transmit data to individual receivers according to them. This approach of apportioning r also gives the reader an intuitive picture on how "buffer sharing" is achieved even though the clients are not physically located at the same place. We call these operations and methods of sharing the communication channel "Lossless Video Aggregation".

Chapter 3

Minimization of Buffer Size

In the last chapter, we set up the two-phase architecture for LVAS and state the buffer minimization problem with two constraints. It turns out that the second-phase problem is hard to solve optimally. A simple idea is to try to equalize the occupancy levels of all receiver buffers. Specifically, to schedule the transmission of individual video streams, the total buffer occupancy (which is derived from the total display and receiving curves during the first phase of the algorithm) is divided into equal portions and assigned as the buffer occupancies of individual receivers. From the display and buffer-occupancy curves of each video stream, we can then derive its receiving, and therefore transmission, curve. This method, however, does not work. Figure. 3.1 is an example to illustrate this point. Here, two streams are multiplexed together. The stacked bars represent the magnified version of the display curve. At time slot $t+1$, client 1 has a large amount of data to be displayed while client 2 only has little. If we simply divide the buffer occupancy into two halves, client 2's arrival curve, which equals to the sum of its display curve plus its buffer occupancy at that time, would be

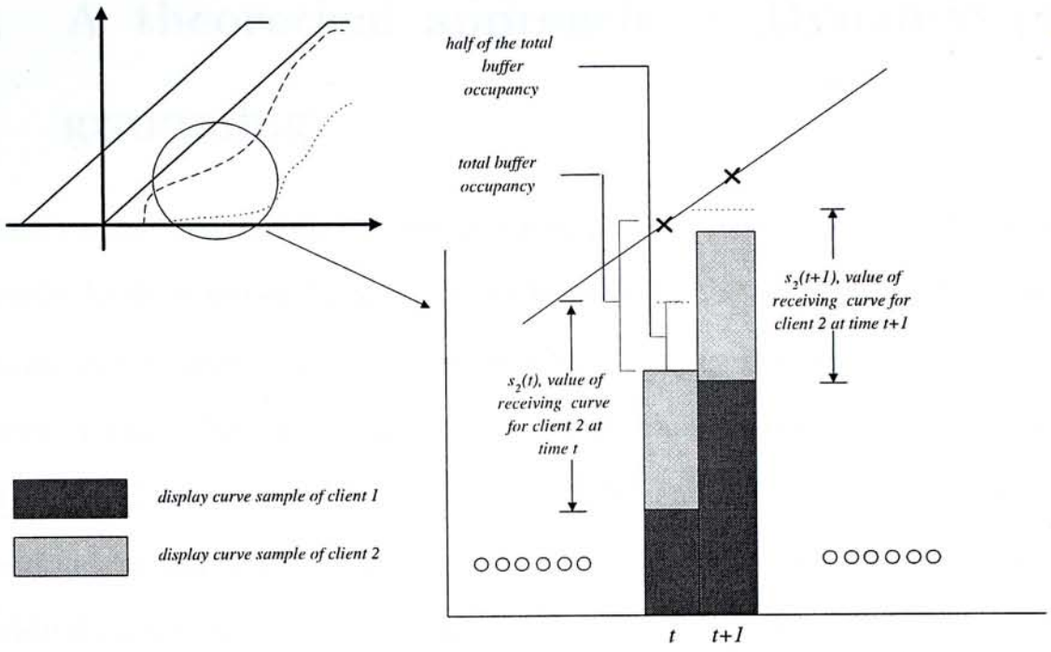

Figure 3.1: An example showing that a simple division of the buffer is not a feasible solution to the buffer minimization problem

decreasing from time t to time $t+1$, which of course is physically impossible. Therefore, the method of simply dividing total buffer occupancies into two equal halfs does not yield a feasible solution.

The general buffer minimization can be formulated as a dynamic programming problem and be solved theoretically. However, the state space is too large for this approach to be practical. We shall propose a heuristic which can solve this problem with a much lower order of complexity without sacrificing much of the optimality. The dynamic programming and heuristic approaches are discussed in the next two sections.

3.1 A theoretical approach — Dynamic programming

The second part of the solution for minimum receiver buffer size can, in principle, be determined by formulating the problem as a deterministic dynamic program and be represented as a network with stages. If we define L as the total number of time slots required to transmit all the video streams, the network would have $L+1$ stages (The last one is the ending stage): Each time slot corresponds to one stage. A node at stage t corresponds to a set of values of the individual buffer states after a time slot t, and is labeled by a vector

$$\{\vec{q}(t) \triangleq (q(1,t), q(2,t), \ldots, q(n,t)) \quad : \quad q(i,t) \geq 0 \qquad \sum_{i=1}^{n} q(i,t) = s(t) - A(t,\vec{d}) \quad \forall i\} \tag{3.1}$$

where $s(t)$ is the sending curve which has been determined in the global part of the problem and $A(t,\vec{d})$ represents the aggregate display curves. Each state is represented by the buffer occupancies of the n receivers $\vec{q}(t)$. Nodes at adjacent stages are connected by links. The recursive relationship between a state in a stage and an arbitrary state in the next stage is

$$f(\vec{q}(t)) = min_{\vec{q'}(t+1)}\{D[\vec{q}(t), \vec{q'}(t+1)]\}, \tag{3.2}$$

where

$$D[\vec{q}(t), \vec{q'}(t+1)] \triangleq \begin{cases} \infty & : \quad A_i(t+1, d_i) + q'(i,t+1) < A_i(t,d_i) + q(i,t) \quad \exists i \in [1,\ldots,n] \\ \max\{q(1,t), q(2,t), \ldots & \\ q(n,t), f(\vec{q'}(t+1))\} & : \quad \text{otherwise} \end{cases}$$

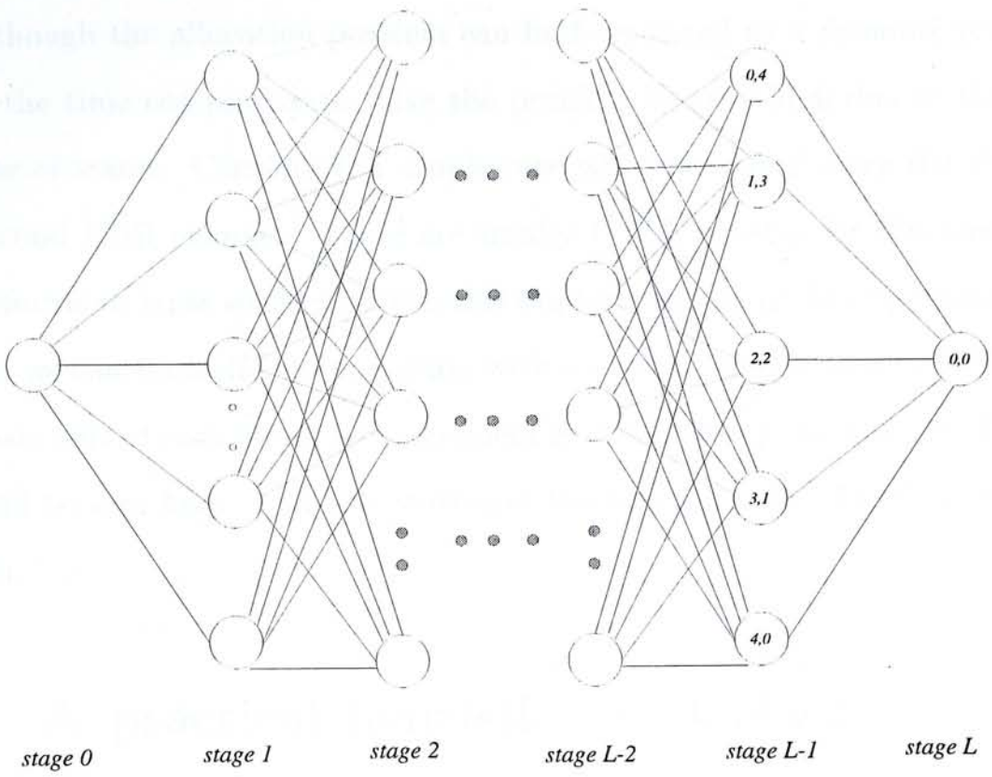

Figure 3.2: A typical dynamic-programming network for two streams. $s(L-1) - A(L-1, \vec{d}) = 4$ units of bits. Thus there are 5 states in the $L-1$ stage.

$D(.)$ is the largest buffer size among the n clients for state $\vec{q}(t)$ from time t through state $\vec{q}'(t+1)$ to the end of the transmission. The infinity in the first case is only to screen out those impossible paths, which would have resulted in a "decreasing" cumulative sending curve. $f(\vec{q}(t))$ is the minimum buffer size, given the current buffer state is $\vec{q}(t)$. Fig. 3.2 shows an example that corresponds to the aggregation of two video streams.

Based on this network, the task is to find $f(\{0, \ldots, 0\})$, which corresponds to a path from stage 0 to stage L such that the maximum receiver buffer size is minimized. This type of program can be solved by the conventional *forward dynamic programming.*

Although the allocation problem can be formulated as a *dynamic programming*, the time complexity to solve the problem is quite high due to the huge number of states. Consider the simple case where 8 clients share the 10Mbits per second CBR channel. There are totally $\binom{10M+7}{7}$ states for this time slot. Such enormous state space prohibits this implementation in the experiment. Although we can tradeoff the complexity with a coarser buffer states (for example, the basic unit of each buffer size increment is 100 kilobits), the number of states will still be very high. And this motivates the heuristics introduced in the next section.

3.2 A practical heuristic — Backward Equalization

At the beginning of this chapter, we have considered an example of a simple method to partition the bandwidth by equalizing the buffer occupancies. We also point out why it does not guarantee to work in general. In this section, we propose a heuristic that performs buffer-occupancy equalization *backward* in time. The advantage of doing so is that when we come across a very large frame at time t, we can use the time slots before t to smooth out the burst. The display curve would not be decreasing using this method.

The dynamics of a specific stream i can be described as follows.

$$B_i(t+1) = B_i(t) - X_i(t-d_i) + \delta s_i(t) \tag{3.3}$$

, where $\delta s_i(t) \triangleq s_i(t+1) - s_i(t)$. It basically states that the buffer occupancy at time $t+1$ equals that of the previous time slot minus the bits consumed plus

the bits received. Since $X_i(t-d_i)$ is given and $B_i(t+1)$ is known at time slot $t+1$, our goal is to adjust the $\delta s_i(t)$ such that $B_i(t)$ for all i are approximately equal, hence minimizing the maximum buffer occupancy among the streams. Furthermore, buffer size must be greater than or equal to zero at any time in order to be feasible, i.e., $B_i(t) \geq 0 \quad \forall t$. These goals can be represented as follows.

$$\min \max_i B_i(t) \tag{3.4}$$

where

$$\begin{aligned} B_i(t) &= B_i(t+1) + X_i(t-d_i) - \delta s_i(t) \geq 0 \\ \sum_i \delta s_i(t) &= \begin{cases} r & : \quad t < t'_n \\ 0 & : \quad t \geq t'_n \end{cases} \end{aligned} \tag{3.5}$$

Let us see how the heuristic works. For easy explanation, we fix all the startup delay to be the same (i.e., $d_i = d$) and each sequence has the same length L. When all clients have finished displaying, their buffer occupancies should be zero, i.e. $B_i(L+d) = 0$. For the time slots in $t = t'_n$ (the time when all data for all streams have been transmitted, refer to equation 2.7) to $t = d + L$, nothing needs to be sent. Hence, $s_i(t) = 0, \forall i$. The operation for each time slot is simply to add $X_i(t-d_i)$ to $B_i(t)$. For those time slots from 0 to t'_n, the operations are as follows: After adding $X_i(t-d_i)$ to it, the n $B_i(t)$ are sorted in descending order. The r bits are then distributed to the stream with the largest $B_i(t)$ until its value equals the second largest $B_i(t)$. If there are any bits remaining, they will be distributed to these two streams until their $B_i(t)$ equals the third largest $B_i(t)$. This operation is repeated until either all the r bits are used up or all

the $B_i(t)$ are equal. In the latter, the remaining bits are equally shared by the n sequences. The idea is to reduce the maximum differences of the buffer occupancies. These operations are run from a time slot to another backward in time. The buffer size requirement $\max_{i,t} B_i(t)$ is updated at the end of each time slot. Since this method tries to equalize the buffer occupancies of each stream, the required buffer size should be reasonably small, compared with individual transmission through different CBR channels.

As a concrete example to the above procedures, suppose that $n = 4, r = 8$ and the initial $B_i(t) - X(t - d_i)$ after sorting is $10, 8, 5, 4$ units respectively. At the first round, 2 units will be given to the first sequence and the $B_i(t)$ becomes 8,8,5,4. Then 3 units each are given to both the first and second sequence. The updated $B_i(t)$ is now 5,5,5,4. Because the r bits has been used up, the procedure stops here for this time slot. Therefore $B_i(t-1) = 5,5,5,4$ and $\delta s_i(t) = 5, 3, 0, 0$. In other words, the first stream gets 5 bits and the second stream get 3 bits in this time slot. The operations continue until $B_i(0)$ is found.

This heuristic is not optimal. It does not consider the time slot from $t = 0$ onwards to the current time slot when determining the current buffer state. However, it tries to equalize the buffer occupancies in each time slot so that they will not become too large when large frames are encountered. In real world, however, this heuristic is quite enough to provide a reasonable buffer size reduction, compared with individual transmission of different streams. In next section I will present some experimental data to demonstrate the closeness to optimality of this heuristic.

3.3 Simulation results of the heuristic method

The experiments indicates that very significant receiver-buffer reduction can be achieved with *lossless aggregation.* The heuristic algorithm described in the preceding section was used. We used a trace of MPEG1 video from Bellcore [12] to conduct the experiments. The trace recorded the frame sizes of approximately two hours of the movie *Star Wars.* The frame rate is 24 frames per second. The digitized frame size is 480×504. Each frame is divided into 30 *slices.* This trace was used to generate 16 "artificial" traces for experiments. An artificial trace was constructed by concatenating eight pieces of 15-minute segments, each of which was extracted from *Star Wars* with a random starting point. That is, to generate a 15-minute segment, we randomly and independently chose a starting point s within the two-hour movie, and the interval between s and s plus 15 minutes formed a 15-minute segment. Figure 3.3 shows two of the 16 traces.

The experiments explore the extent to which buffer reduction can be achieved as a function of the number of streams n being aggregated. When $n = 1$, we have the non-aggregated case, in which we randomly chose one of the 16 traces. For $n = 2$, in addition to the already chosen trace for $n = 1$, we randomly chose another trace out of the remaining 15 traces and performed the aggregation. We did this repeatedly and increased n until all 16 traces had been aggregated. This forms a trace-selection pattern. Based on this trace-selection pattern, for $n = 1, 2, \ldots, 16$, the buffer size required using the above aggregation heuristic is compared with that of using the conventional method of sending separate streams over separate CBR channels.

In practice, each receiver (e.g., in a set-top box) is equipped with a fixed

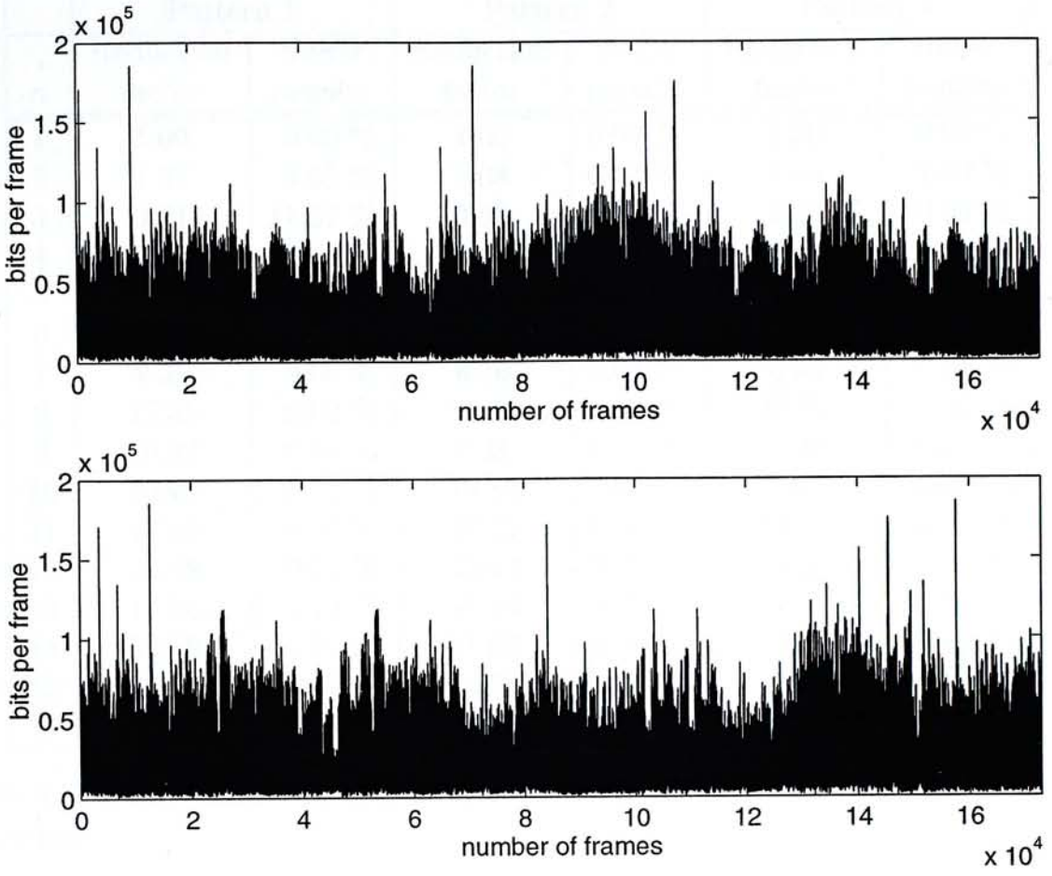

Figure 3.3: Bit rate profiles of two 'artificial' sequences

amount of memory. The amount of memory needed should be set by testing a wide variety of movies (videos) and choosing the maximum buffer required. That is, the worst-case buffer requirement should be the benchmark for setting the memory size. For this reason, to interpret our experimental results, the maximum buffer required among all streams is used as the measure for comparison between aggregation and the conventional methods.

The *buffer reduction factor* is defined to be the value of $max_{i,t}B_i(t)$ without aggregation over that with aggregation. The buffer reduction for three trace-selection patterns are shown in Table 1. The general trend is that the more streams are aggregated, the larger is the buffer reduction. As shown in the table,

	Pattern 1		Pattern 2		Pattern 3	
n	Reduction factor	Buffer penalty	Reduction factor	Buffer penalty	Reduction factor	Buffer penalty
1	1.00	0.00 %	1.00	0.00 %	1.00	0.00 %
2	1.87	3.65 %	5.08	0.02 %	1.69	7.89 %
3	3.31	11.57 %	3.32	0.01 %	6.12	0.02 %
4	5.46	0.02 %	8.63	0.04 %	3.40	21.98 %
5	5.44	0.03 %	9.18	0.05 %	6.90	0.03 %
6	10.50	0.04 %	7.27	0.04 %	5.67	0.01 %
7	9.39	0.06 %	8.16	0.04 %	5.56	0.02 %
8	13.35	0.02 %	11.33	0.03 %	10.87	0.05 %
9	10.97	0.04 %	8.21	0.02 %	8.60	0.02 %
10	19.04	0.12 %	15.83	0.06 %	9.62	0.03 %
11	17.02	0.01 %	15.71	0.06 %	17.17	0.06 %
12	21.09	0.12 %	13.42	0.00 %	17.20	0.04 %
13	17.56	8.14 %	16.34	0.09 %	15.49	0.03 %
14	26.59	0.03 %	21.02	0.06 %	17.53	0.04 %
15	24.04	0.02 %	21.98	0.08 %	21.60	0.09 %
16	23.70	0.01 %	23.70	0.03 %	23.70	0.01 %

Table 3.1: buffer reduction factor and buffer penalty of a heuristic aggregation algorithm

a buffer reduction factor of more than 20 can be achieved using aggregation when $n = 16$.

Another question is how good the solution of the heuristic is when compared with the optimal solution. Recall that the optimal solution is difficult to solve, making it difficult for us to perform a direct comparison. However, we can evaluate the goodness of the heuristic indirectly, as described below.

It turns out that if we were to relax the original optimization problem by dropping *Constraint 2* (refer to the end of Chapter 2), the optimization problem would be easy to solve, although the fact that $s_i(t)$ could decrease with t is non-physical. The modified optimization problem can be solved simply by dividing the global buffer occupancy $B(t, \vec{d})$ found in the first phase by n and assign the

same amount of buffer occupancy to each stream. That is, $B_i(t, d_i) = \frac{B(t,\vec{d})}{n}$ for all i.

Let $B_{opt} \triangleq max_{i,t} B_i(t, d_i)$ be the maximum buffer occupancy in the original optimization problem (i.e., with *Constraint 2*) and B'_{opt} be that of the modified problem. It is obvious that $B'_{opt} \leq B_{opt}$ since the modified problem has one fewer constraint. Using the heuristic algorithm we can also get a buffer requirement B_{heu}. The percentage of buffer penalty due to the use of the suboptimal heuristic is $\frac{B_{heu}-B_{opt}}{B_{opt}} \times 100\% \leq \frac{B_{heu}-B'_{opt}}{B'_{opt}} \times 100\%$. The RHS is an over-estimate of the buffer penalty. Based on this estimate, the experimental results (Table. 1) show that the heuristic algorithm is close to optimal. With seven exceptions out of 48 samples [1], the buffer penalty is smaller than 0.1%.

[1] These seven exceptions are due to the non-linearities of the sequences. If a certain combination of sequences makes a large peak on the aggregate display curve $A(t, \vec{d})$, the buffer requirement using aggregation will be larger and, in turn, the reduction factor will be smaller, although it is still far larger than 1.

Chapter 4

Bit-rate allocation in video aggregation with fixed buffer size

4.1 Problem formulation

In the preceding chapter, we have shown that the required receiver buffer size can be reduced quite significantly using *lossless aggregation*. There are three aspects of the problem formulation that are noteworthy:

1. The buffer size is a parameter to be optimized (minimized).
2. The channel rate r is not fixed *a priori*; rather, it can be varied and optimized according to the set of videos being tested (refer to phase 1 of the *lossless aggregation* in section 2.2).
3. The channel rate r, once set, is to be fully utilized.

While the formulation is good for the investigation of the buffer reduction using aggregation, the algorithm therein cannot be applied directly in many

practical situations for dynamic determination of the transmission schedules of video streams.

First, the buffer size at the receivers may not be determined *a priori* in practice: for example, once the memory in the set-top box is fixed, it is fixed forever. Second, the video server may lease from a network operator a fixed amount of channel rate r for the transmission of the video streams: hence, r is fixed. Third, given that an amount of bit rate r has been leased, it is up to the video server to transmit at a rate lower than r: for example, the server may choose to do so if transmitting at the full rate r would lead to buffer overflow at the receivers.

This chapter looks at the aggregation problem from a different angle: *Given a CBR channel with rate r and a fixed receiver-buffer size of B at all the receivers, how should the transmission of a group of video be scheduled so that receiver underflow and overflow would not occur in any of the receivers?*

Consider the transmission of data from the server to the clients. With reference to Fig. 4.1, we see that the following global constraints must be satisfied:

$$\begin{array}{lcl} \sum_{i=1}^{n} A_i(t, d_i) & \leq & \sum_{i=1}^{n} s_i(t) \leq \sum_{i=1}^{n} A_i(t, d_i) + N \times B \\ \sum_{i=1}^{n} \delta s_i(t) & = & \sum_{i=1}^{n} [s_i(t+1) - s_i(t)] \leq r \end{array} \qquad \forall t \qquad (4.1)$$

The first constraint is necessary so that receiver overflow and underflow do not occur. The second constraint is for ensuring that the total bit rate used is smaller than the channel rate.

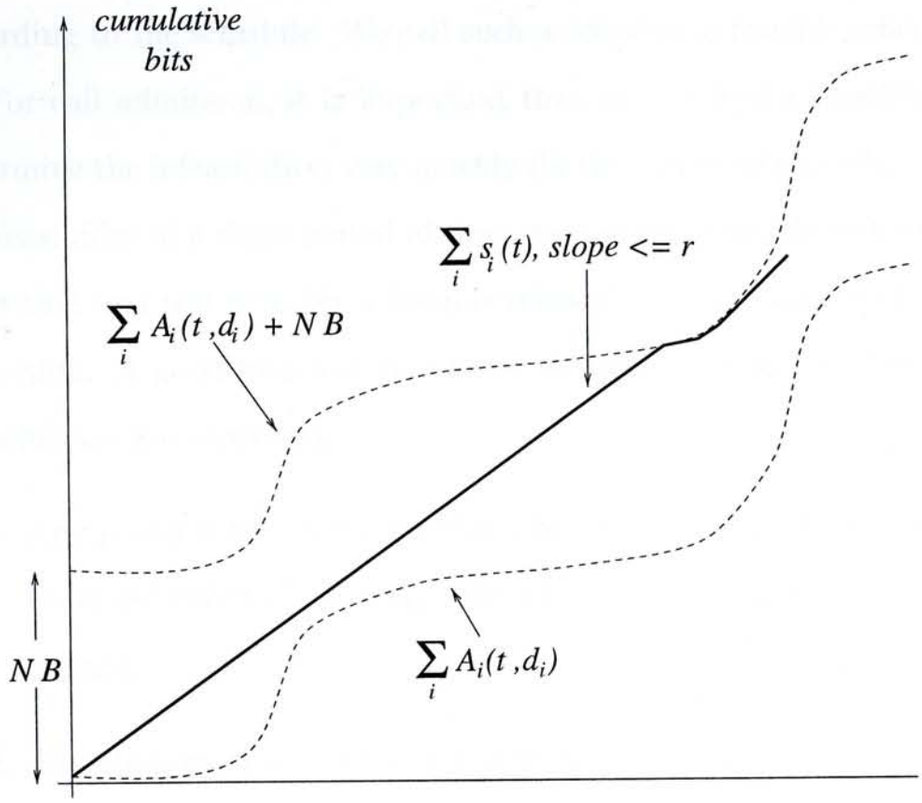

Figure 4.1: Aggregation of a bundle of videos (Note that the aggregate receiving curve $\sum_i s_i(t)$ is bounded between $\sum_i A_i(t, d_i)$ and $\sum_i A_i(t, d_i) + N \times B$. The slope is bounded above by r.)

In addition to constraints (4.1), we also have the following constraints for the prevention of buffer overflow and underflow at individual receivers.

$$\begin{aligned} & A_i(t, d_i) \leq s_i(t) \leq A_i(t, d_i) + B \\ & s_i(t) \leq s_i(t+1) \end{aligned} \qquad \forall t, 1 \leq i \leq n \tag{4.2}$$

The first equation of constraint (4.1) can be derived from that of constraint (4.2). Hence, there are altogether only three independent equations.

The bit-rate allocation problem is stated as follows. Given a set of videos and their display curves, is it possible for the server to schedule the transmission to individual clients $\{s_i(t) \forall i, t\}$ such that the above constraints are satisfied? If the answer to the question is yes, this set of sequences can then be sent out

according to the schedule. We call such a schedule a feasible schedule.

For call admission, it is important that we can find a feasible schedule (or determine the infeasibility) very quickly (in the matter of seconds). To determine the feasibility in a short period of time, we can use a simple heuristic scheduling algorithm and test whether a feasible schedule can be found with this heuristic algorithm. A good heuristic should be characterized by two features as far as its solutions are concerned:

1. Any problem to which a feasible schedule cannot be found with the heuristic is unlikely to have a feasible schedule with any other scheduling algorithms.

2. The chances of accepting new requests in the future are maximized.

To achieve the above, the heuristic should fully utilize the channel rate and the buffering space at the receivers. A good scheduling heuristic should allow the streams to share the channel rate in an intelligent manner. The bit rate should be shared among the streams in a way that the receivers that urgently need more data at a particular time slot will be transmitted more data; by the same token, the receivers with almost full buffer should not be transmitted so much data as to lead to buffer overflow.

An issue is how to determine the relative urgency of data transmissions to the clients. A simple scheduling method is to transmit the data to attempt to equalize the occupancy levels of all receiver buffers. However, this method has some shortcomings, owing to the VBR nature of video: suppose that a particular receiver is about to display a large frame, whereas the other receivers' are about

to display a small frame. If equality is maintained at the receiver buffers, there may not be enough data for the large frame to be displayed at the first receiver. Such kind of bit-rate allocation does not take the future frame characteristics into account.

Section 4.2 considers several fast heuristic scheduling algorithms. They can mainly be divided into two categories: Fixed capacity assignments and *lossless aggregations*. The former category assigns a certain amount of bandwidth to each sequence, regardless of the variable-bit-rate property of the videos. The latter takes advantage of the bit-rate fluctuations of the videos and can schedule them in a more efficient manner.

To implement LVAS in practice, the scheduling algorithm should be very fast in order to process the video requests in real time. A point sampling technique is evaluated in section 4.3 to speed up the aggregation process. The trade-off accompanied with the speed-up will also be discussed.

Simulations of different scheduling methods as well as the point sampling techniques were presented at section 4.4. We can see that the aggregation methods out-perform those fixed-capacity assignments very much. It demonstrates the effectiveness of LVAS. The point sampling method significantly reduces the computation complexity of the aggregation, with the cost of some receiver buffer reservation. As the effect of buffer size to the multiplexing gain is small when the buffer is reasonable large (this is one of findings in the experiments), the point sampling method can be used in practice to speed up the aggregation process.

4.2 Different bit-rate scheduling methods

A. Fixed-Capacity Assignments

Divide-by-n Method

When there are n streams, the divide-by-n method simply divides the CBR channel rate r into n CBR subchannels, each with CBR rate $\frac{r}{n}$. For each stream, the server keeps track of the receiver buffer occupancy based on its knowledge of the display curve and its transmission schedule to the receiver. When the receiver buffer is not full, the stream will be transmitted at rate $\frac{r}{n}$. Otherwise, the stream will be transmitted at rate just enough to replenish the consumed (displayed) bits in the buffer in each time slot.

This method is simple and its time complexity is small. The operation only involves the buffer occupancy updating. The average number of the updating operations per stream is l, where l is the mean length (in number of frames) of the n videos. Furthermore, since there is no coordination among sequences, the n streams can be checked in a parallel way.

The drawback of this heuristic is that efficient "resource sharing" cannot be achieved. When the buffer of a stream is full, another stream with a relatively low buffer occupancy level cannot make use of the unused bit rate to acquire more data, making underflow more likely in the future.

Mean proportional method

The mean proportional method is a simple extension of the divide-by-n method. Instead of dividing the channel rate evenly, we divide it according to the videos' mean bit rates. Let ρ_i be the mean bit rate of stream i over the

whole sequence. Then, the bit rate assigned to a particular stream j is $r\rho_j / \sum_i \rho_i$. Since the mean bit-rate can be calculated off-line for the stored video, the time complexity is the same as the divide-by-n method.

The mean proportional method should be better than the simple divide-by-n method when the difference between the bit-rate of the streams is large. It is reasonable to allocate more bandwidth to a sequence that has more data to send. However, the difference in performance compared with the divide-by-n method is difficult to predict when the bit-rate averages of the sequences are in the same order of magnitude.

Variance proportional method

The function of the receive buffer is to pre-fetch the data before the bursts. However, if the variation of the bit rate is very large, the buffer is not capable or does not have time to download enough data beforehand. This would result in underflow. To prevent such kind of situations from happening, the bandwidth can be partitioned according to the bit-rate variance of the sequences.

Instead of mean or variance, other statistical indicators or combinations of them can also be used. However, the performances of these fixed bit-rate allocation in general is not as satisfactory as the more dynamic bit-rate allocation schemes described in the next subsection.

B. Aggregation Methods

Partial look-ahead scheduling

The partial look-ahead scheduling scheme is a very general dynamic bit-rate allocation scheme. The assigned bit rate to a stream varies from slot to slot. To describe this scheme, let us first define some notation (Fig. 4.2). Consider

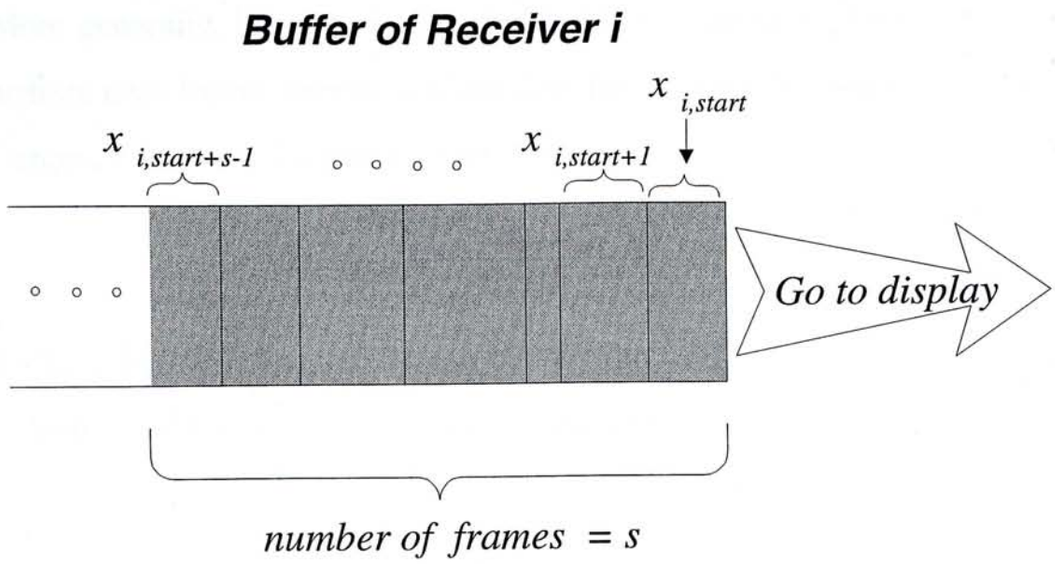

Figure 4.2: Notation for describing the buffer states for look-ahead scheduling

a particular stream. Denote the size of frame j of this stream by x_j and the number of *complete* frames stored in the receiver buffer by c. Let b be the number of bits of the partially filled frame, if any, at the end of the receiver buffer. Let the first frame currently in the buffer be indexed by *start*. Then, the index of the last complete frame in the buffer is $start+c-1$. For each client, we can define an *urgency measure* that describes how urgently the receiver buffer needs more data from the transmitter:

$$\frac{x_{start+c} - b}{c} \tag{4.3}$$

The motivation for this definition is as follows. In c time-slots, the currently partially-received frame will need to be displayed at the receiver. This means that the server must transmit $x_{start+c} - b$ bits to the client in c frame times to prevent underflow. This urgency measure is simply the average rate at which the transmission must occur.

More generally, instead of just considering the urgency of transmitting the immediate next frame, we can also consider frames more advanced in the future. The urgency measure for frame $start + c + k$ is

$$\frac{\sum_{i=0}^{k} x_{start+c+i} - b}{c + k} \tag{4.4}$$

By looking ahead w frames, we can choose the worst-case urgency measure as an indication of the urgency to transmit data to the client:

$$\max_{0 \le k \le w-1} \frac{\sum_{i=0}^{k} x_{start+c+i} - b}{c + k} \tag{4.5}$$

Let U_m be the urgency measure of client m. A bit-rate allocation algorithm for the server is to transmit to the client that has the largest U_m. The detailed algorithm is as follows. For each time slot, the following steps are performed repeatedly:

- *Calculate the urgency measures*: the urgency measure of each stream is computed according to (4.5).
- *Select the winner*: the server finds the stream with the largest urgency measure.
- *Grant the bits to the winner*: The number of bits that will be sent for the winning stream is

$$max \quad (x_{start+c} - b, \quad \textit{the shared bits remaining for this time slot,} \quad \textit{unfilled buffer size of this client})$$

These three operations are performed repeatedly until no bandwidth is available for this time slot or all the receivers' buffers are filled up. Then the server waits until the next time slot, when upon the same steps are repeated.

Let n be the number of streams and l be the average number of frames in each stream. Then nl is the total number of frames that must be transmitted. The number of divisions used to find each urgency measure U_m is w. Each loop of the above steps requires the computation of n urgency measures, or n divisions. If we assume that to transmit one frame (in any stream) we have to go through the above loop once and only once, the number of divisions used in l time slots is $O(nl \times nw) = O(n^2wl)$.

It turns out that in practice w need not be very large in order to prevent most of the underflow. In fact, the complex computations prevent a large w in practice. In Section 4.4 we will show that the performance of this heuristic is already satisfactory even when $w = 1$.

Frame equalization scheduling

The frame equalization scheduling method is a very simple dynamic bit-rate allocation scheme. The main concept is to keep the numbers of frames in the receiver buffers as equal as possible. In each time slot, r bits are allocated to the n streams in a round-robin fashion. Each time slot may contain less or more than one round of bit allocation. The transmitter attempts to transmit one frame for each stream in each round, regardless of the frame size. The operation stops and waits for the next time slot when either one of the receivers is filled up or no bandwidth is left for this time slot. At the end of a time slot, a frame can be partially sent for either of the above reason. If a frame is partially sent

because of no bandwidth for this time slot, the bits of the next time slot will be distributed starting from the remaining part of this frame.

A problem of this method occurs when the frames of a sequence is so large that the associated buffer is filled up, while others are still only partially filled. Although there is still capacity for more frames, the receivers that are not full cannot get more since the number of frames in each buffer must be equal. A modified scheme is to use the bandwidth more aggressively: when some clients have their buffer filled up, the round-robin scheme described in the pervious paragraph will not stop. It continues with those sequences that still have empty buffer space. The operations stop only when all the bits for this time slot are used up or all buffers are full. Like the original scheme, those sequences that have partially sent frames would send first in the next time slot.

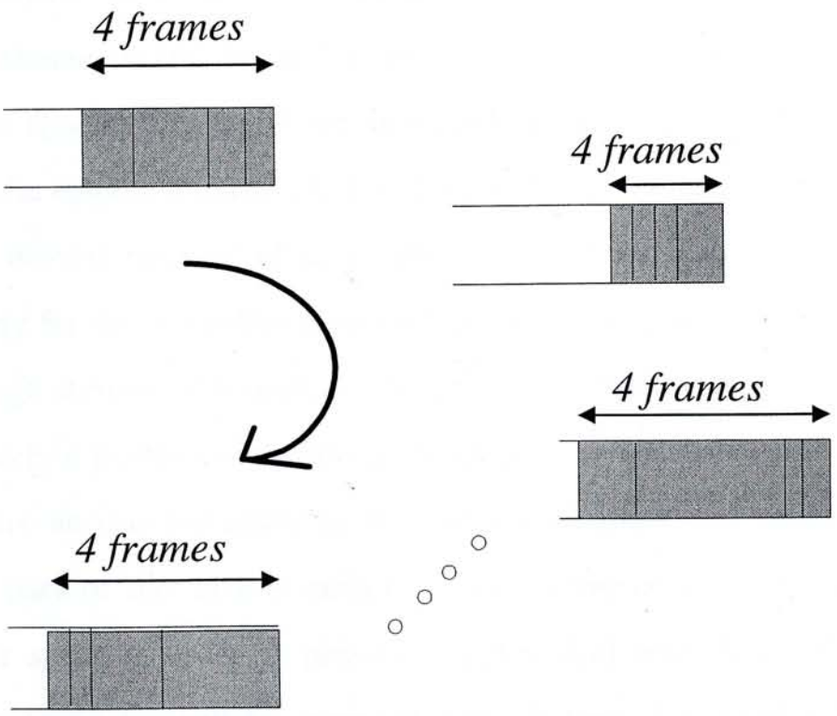

Figure 4.3: The concept of the frame equalization method

The whole scheme depends on the assumption that the decoding time of a frame is constant for all decoders. This is quite reasonable for current video technologies. In fact, the decoding time should be tightly bounded in order not to cause annoyance to the audience.

The time complexity is rather small using this method. The total number of bandwidth (remaining bits) updating among all n streams is $n \times l$. Since this method tries to keep the numbers of frames in the buffers equal, the backlogs of receiver buffers measured in *time* (as opposed to bits) are approximately equal. This minimizes the chance of underflow of any stream.

4.3 Speed up using point sampling technique

One characteristic of a good scheduling algorithm is fast computation. A call admission scheme should be as fast as possible in order to process real-time requests. To shorten the searching time for feasible schedules, we can use only samples of the display curves, $\{A_i(t \times \Delta, d_i \times \Delta), \Delta > 1 \in \mathcal{Z}^+\}$ (the samples of the display curves), instead of all points on the curves, $\{A_i(t, d_i)\}$. The order of complexity for the heuristics described in the previous section is proportional to the average number of frames, l. The processing time would be substantially reduced if only a portion of points on the display curves are examined.

Attractive as this technique is, it may impose some cost on the receivers. Since only part of the data is examined, each receiver needs to reserve some extra buffer space in order to prevent overflow and underflow. Furthermore, a larger startup delay will be expected since Δ time-slots of bits in extra are needed to be preloaded beforehand.

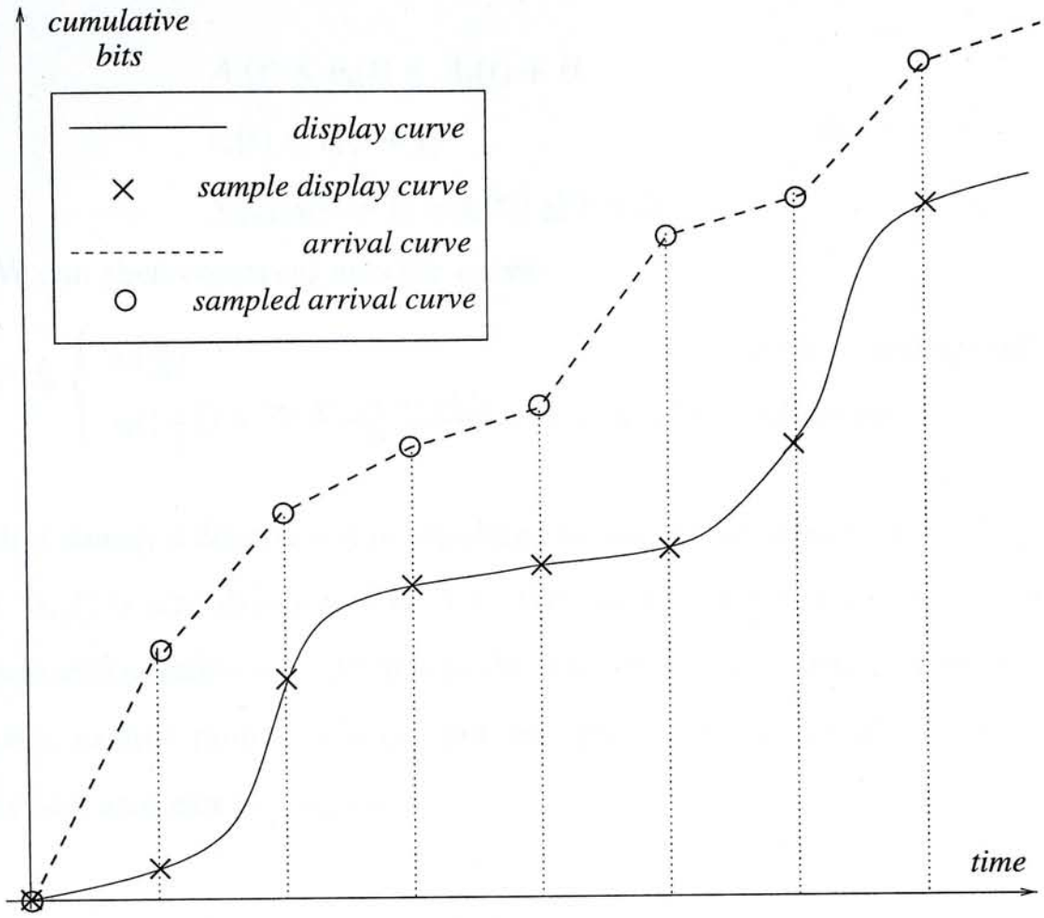

Figure 4.4: A display curve and the corresponding sampled points

Let us look into this point sampling idea in detail. To simplify the notation, the dependency of startup delay in $A_i(t, d_i)$ is dropped. Fig 4.4 shows an example display curve and the corresponding sample points. Applying one of the previously described scheduling methods to the sampled points $\tilde{A}_i(t, d_i) \triangleq A_i(t \times \Delta, d_i \times \Delta)$ using $r \times \Delta$ bits per sample (the time difference between two sample points are Δ frames), we can find a feasible schedule $\tilde{s}_i(t)$ (by equations 4.1,4.2):

$$\begin{aligned} &\tilde{A}_i(t) \le \tilde{s}_i(t) \le \tilde{A}_i(t) + B \\ &\tilde{s}_i(t) \le \tilde{s}_i(t+1) \qquad\qquad \forall t \\ &\textstyle\sum_{i=1}^{n} [\tilde{s}_i(t+1) - \tilde{s}_i(t)] \le r \times \Delta \end{aligned} \tag{4.6}$$

We can then construct another curve

$$\hat{s}_i(t) \triangleq \begin{cases} \tilde{s}_i(\frac{t}{\Delta}) & \text{, if } t \text{ is a multiple of } \Delta \\ \tilde{s}_i(\lfloor \frac{t}{\Delta} \rfloor) + \frac{\tilde{s}_i(\lfloor \frac{t}{\Delta} \rfloor + 1) - \tilde{s}_i(\lfloor \frac{t}{\Delta} \rfloor)}{\Delta} \times (t - \Delta \lfloor \frac{t}{\Delta} \rfloor) & \text{, otherwise} \end{cases} \tag{4.7}$$

which is simply a dilated and interpolated version of the sampled receiving curve $\tilde{s}_i(t)$. $\hat{s}_i(t)$ is also drawn in Fig. 4.4. We can see that using $\tilde{s}$ as the sending curve is still possible to underflow in the case that a burst of large frames occurs between sample points. To prevent any possibility of underflow, the actual receiving curve can be defined as

$$s_i(t) \triangleq \hat{s}_i(t + \Delta) \tag{4.8}$$

We can show that the total required bit-rate would not exceed the channel bandwidth r in any time slot. The sum of bit-rate needed in time slot $t = t'\Delta + \alpha \quad (0 \le \alpha < \Delta)$ is

$$\begin{aligned} \sum_{i=1}^{n} [s_i(t+1) - s_i(t)] &= \sum_{i=1}^{n} \{\hat{s}_i(t + \Delta + 1) - \hat{s}_i(t + \Delta)\} \\ &= \sum_{i=1}^{n} \{\hat{s}_i[(t'+1)\Delta + \alpha + 1] - \hat{s}_i[(t'+1)\Delta + \alpha]\} \\ &= \sum_{i=1}^{n} \{[\tilde{s}_i(t'+1) + \frac{\tilde{s}_i(t'+2) - \tilde{s}_i(t'+1)}{\Delta} \times (\alpha + 1)] \\ &\quad - [\tilde{s}_i(t'+1) + \frac{\tilde{s}_i(t'+2) - \tilde{s}_i(t'+1)}{\Delta} \times \alpha]\} \\ &= \sum_{i=1}^{n} \frac{\tilde{s}_i(t'+2) - \tilde{s}_i(t'+1)}{\Delta} = \frac{1}{\Delta} \sum_{i=1}^{n} [\tilde{s}_i(t'+2) - \tilde{s}_i(t'+1)] \end{aligned}$$

$$\begin{aligned} &\leq \frac{1}{\Delta} \times (r \times \Delta) \qquad \text{(by first equation of 4.6)} \\ &= r \end{aligned}$$

Therefore, it will not violate the second condition of 4.1.

To prove that it would not have any underflow, we need to show that $A_i(t) \leq s_i(t), \forall i, t$. If we, again, represent t as a Δ-modulo format, say $t \triangleq t'\Delta + \alpha \quad (0 \leq \alpha < \Delta)$, then

$$\begin{aligned} A_i(t) = A_i(t'\Delta + \alpha) &\leq A_i[(t'+1)\Delta] \qquad \text{(non-decreasing property of } A_i(t)) \\ &= \tilde{A}_i(t'+1) \\ &\leq \tilde{s}_i(t'+1) \qquad \text{(by second equation of 4.6)} \\ &= \hat{s}_i(t'\Delta + \Delta) = \hat{s}_i(t'\Delta + \alpha + (\Delta - \alpha)) \\ &= \hat{s}_i(t + (\Delta - \alpha)) \\ &\leq \hat{s}_i(t + \Delta) = s_i(t) \end{aligned}$$

Therefore this setting of $s_i(t)$ would not cause any underflow problem as long as the sampled receiving curve yields a feasible solution. The extra buffer required to prevent overflow, *compared to the sampled receiving curve*, at time $t = t'\Delta + \alpha \quad (0 \leq \alpha < \Delta)$ is not larger than

$$\begin{aligned} s_i(t'\Delta + \alpha) - \tilde{s}_i(t') &\leq s_i(t'\Delta + \Delta) - \tilde{s}_i(t') \\ &= \hat{s}_i(t'\Delta + 2\Delta) - \tilde{s}_i(t') \\ &\leq \tilde{s}_i(t'+2) - \tilde{s}_i(t') \end{aligned} \tag{4.9}$$

That means the extra buffer needed is no more than the difference between 3 consecutive receiving curve samples in order to guarantee that no overflow would occur. This result implies that the amount of extra buffer needed could only

be determined after the receiving curve samples are calculated. In practice, the reasonable procedure should be to reserve fixed amount of buffer for this sampling penalty. Therefore we need an estimate of the extra receiver buffer which is not in terms of the undetermined sample receiving curve.

An observation can help us find a quite accurate estimate for the extra receiving buffer: the slope of the receiving curve usually would not be larger than that of the display curve (except when the buffers are filling up due to excess bandwidth) since the display curve is smoothed by the client buffer as to be a receiving curve. Therefore the extra buffer required usually would not be greater than the difference between three consecutive *display* curve samples, that is,

$$\begin{aligned} s_i(t'\Delta + \alpha) - \tilde{s}_i(t') &\leq \tilde{s}_i(t'+2) - \tilde{s}_i(t') \\ &\approx \tilde{A}_i(t'+2) - \tilde{A}_i(t') \end{aligned} \tag{4.10}$$

The operations can be like this: Each stream calculates the maximum difference between 3 consecutive display curve samples as the sampling penalty in an off-line manner. Then they submit the display curve and the usable buffer size, which is the real buffer size subtracted by the above sampling penalty, to the server. After that, the server sends back the sampled receiving curve. Each stream performs the dilations and interpolations independently and then begins the transmission.

As equation 4.8 implies, the receiving curves have the same shape as the dilated and interpolated version of the sampled receiving curves, with a horizontal shift of Δ time slots to the left. Therefore the startup delay would increase Δ time slots in order to guarantee that no underflow would occur.

Readers should note that equation 4.9 only tell us the extra amount of buffer

needed for the interpolated curve, compared with that for the *sampled* receiving curve. It is not directly related to the buffer required in the original *non-sampled* case. Therefore the bound of extra buffer needed compared with that of not using sampling remains unknown. Some experiments have been conducted in next section to investigate this matter.

4.4 Simulation results

Different bit-rate allocation schemes

To investigate the performance of the schemes discussed in Section 4.2, we have performed experiments using the same set of streams in section 3.3. The 16 streams were randomly ordered in four ways: this formed four sets of 16 ordered streams for four sets of experiments.

In each set of experiments, all the bit-rate allocation methods discussed above were tested. We investigated the number of streams that can be supported with fixed bit-rate r. The 16 streams were admitted one by one (according to a pre-set order) until the addition of the next stream would lead to underflow. Note that overflow is not a concern because we can prevent it by simply not transmitting data to the receiver when the buffer is about to overflow. The number of streams that can be admitted in this way gives us an indication of how good a bit-rate allocation method is.

The results for different schemes are shown on Fig. 4.5, where the number of accepted sequences is plotted against the channel bit-rate r (in bits per frame). The channel bit-rate ranges from 40kbits per frame to 480kbits per frame, spaced 40kbits per frame apart. The buffer size under tested starts from 5Mbits to

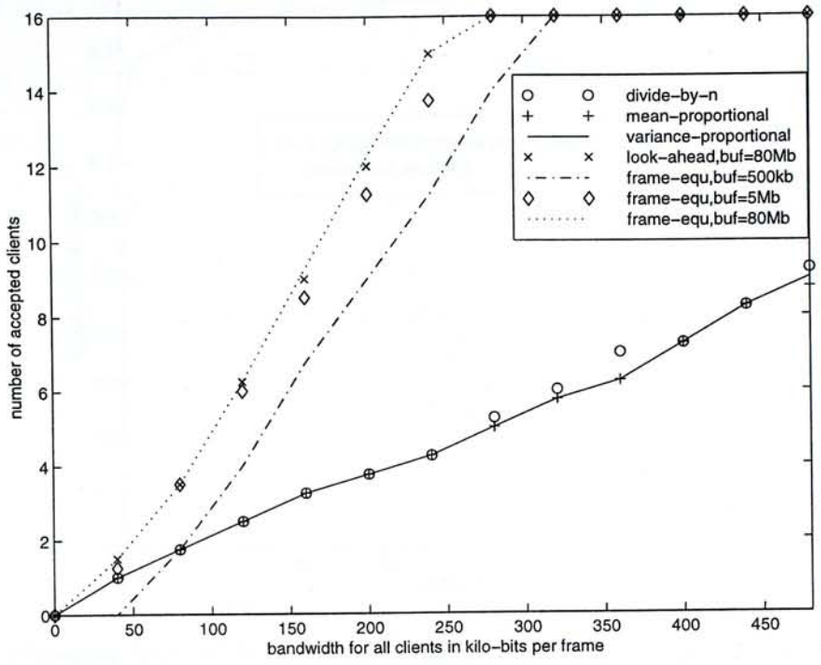

Figure 4.5: Average number of sequences accepted using different scheduling methods

80Mbits, with 5Mbits between samples. The number of accepted sequences in the figure refers to the *average* number of sequences admitted for the four sets of ordered streams.

There is a very significant difference in general between the aggregation schemes and the fixed-capacity assignment methods. All methods under the latter category are similarly bad, and both of the aggregation schemes have roughly the same performance. The number of accepted videos for the fixed-capacity assignments are significantly lower than that of the aggregation methods. For instance, when the bit-rate is 240kbits per frame and the receiver buffer size is 80Mbits, the number of accepted sequences for *partial look-ahead* (window size = 1) is 15 while that of the *variance-proportional method* is only 4.

The fixed-capacity assignment schemes are not sensitive to the receiver buffer

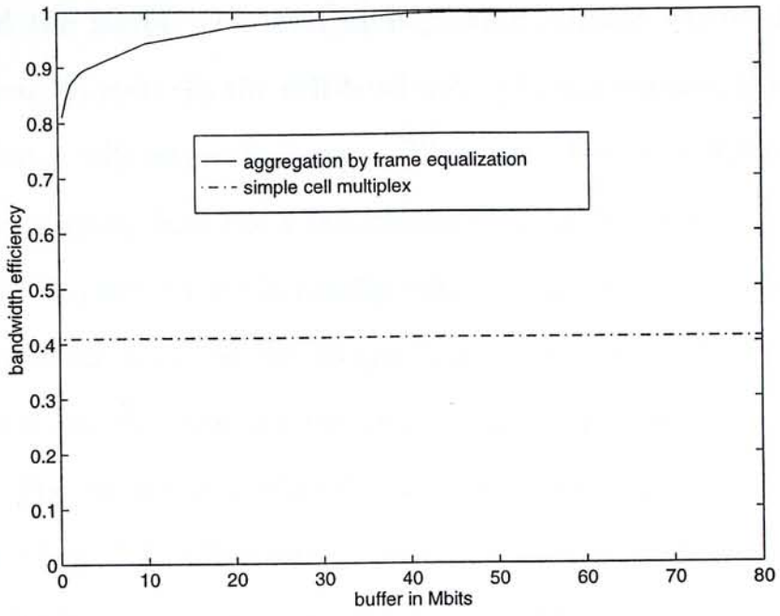

Figure 4.6: Comparison of bandwidth efficiency between *frame equalization* and *cell-level multiplexing* when no cell loss is allowed (lossless transmission)

size at all. The number of accepted streams remains unchanged for any one of them, using buffer size ranging from 5Mbits to 80Mbits. It is because the bandwidth is not efficiently shared in these type of assignment. Even if the buffer is large, it is not fully utilized and therefore the effect of it is not significant.

For the aggregation methods, the size of the receiver buffer do have some effect on the number of supportable streams. Take the example of *frame equalization*, the difference between buffer size = 5Mbits and 80Mbits is considerably small. As long as the buffer in each client is reasonably large (5Mbits in this case), this aggregation method is already able to smooth the VBR burstiness of the videos in a co-operative manner.

The performance of *frame equalization* is the best among all methods. Together with its small computation complexity, it is the best scheme to be used in practice. In Figure 4.6, we compare the bandwidth efficiencies between *frame*

equalization and the simple cell-level multiplexing scheme where the required cell lost rate equals to zero. In the cell-level multiplexing scheme, the video data are packetized into cells at each source. They are then multiplexed together on cell level before going into the CBR channel. To achieve zero cell loss rate, the sum of peak bit-rate, which is totally 600, 409, should be reserved. As the total sum of the mean rates of the 16 streams is 245, 840, the bandwidth efficiency of cell-level multiplexing for the 16 streams with no cell loss is therefore $\frac{245,840}{600,409} = 41\%$. The bandwidth efficiency of *frame equalization* for 16 streams is already 91% when the buffer size is only 5Mbits (625k bytes). It achieves 100% when the buffer size is 40Mbits(5M bytes). The results are indeed quite satisfactory.

From Fig. 4.6, we can see that when the receiver buffer is reasonably large, the bandwidth utilization is rather insensitive to the increase of buffer size. This observation supports the adoption of the point sampling speed-up method, which essentially is a trade-off between the receiver buffer size and the time of the call admission process.

Point sampling method

Figure 4.7 depicts the effects of sampling interval, buffer size and channel rate to the maximum buffer penalty associated, which is calculated by equation 4.9. For clear illustrations, only two values of the sampling intervals are shown in the diagram: $\Delta = 2$ and $\Delta = 20$. We can see that this parameter is a dominant factor to the buffer penalty. It is because as the interval becomes larger, the buffer fluctuations between samples would also become larger and so are the buffer required to smooth out these fluctuations. The buffer size and the

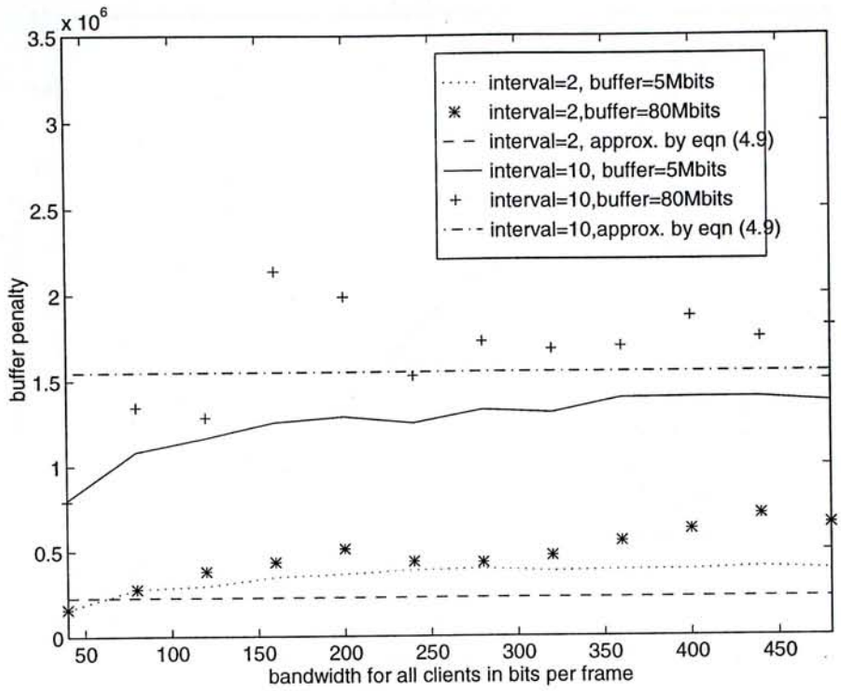

Figure 4.7: Bound of buffer penalty for different sampling intervals

channel rate do affect the buffer penalties. Since our strategy is to fill out the receiver buffer as much as we can, the larger the buffer size and the channel rate, the faster the receiver buffers are filled up, and, in turn, the larger the difference between consecutive receiving curve samples. Figure 4.7 also shows the estimates of buffer penalties using equation 4.10. We can see that the approximations are around the values that is calculated using equation 4.9. It shows that equation 4.10 can be a proper reference to the buffer penalties.

Figure 4.8 compares the number of sequences supported with and without using point-sampling. The real buffer size for the former is the nominal buffer size minus the maximum buffer penalties, calculated by equation 4.9 for all the buffer size under tested. The delay is 20 frame time and the buffer size is 40Mbits for all cases.

We can see that the differences between them is small. There is a little

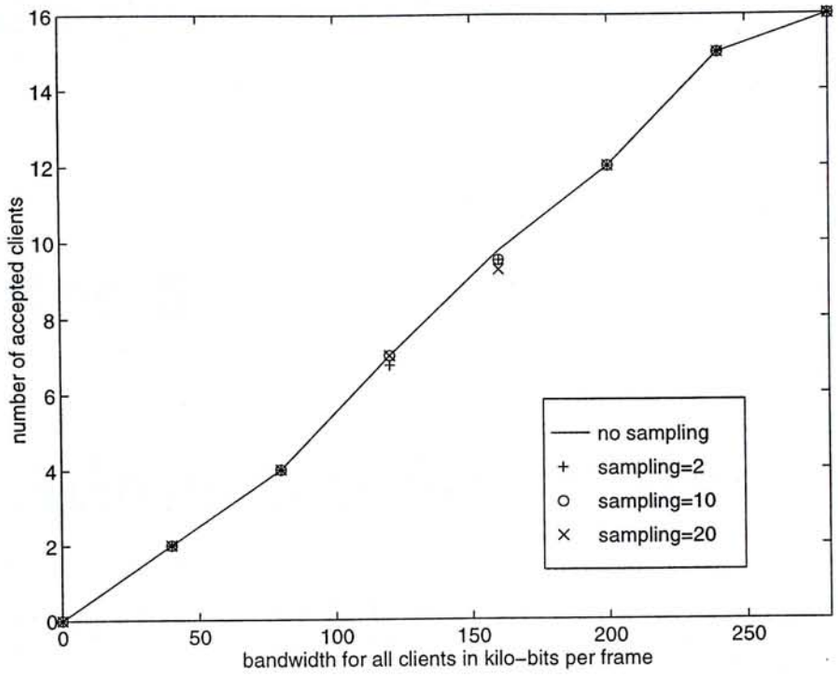

Figure 4.8: Number of supportable streams using frame-equalization, with and without point-sampling

penalty for sampling only when the buffer is small ($5Mbits$). On the other hand, point sampling with sampling interval Δ reduces the computations to approximately one-Δ-th of the original. This fact justifies the usage of point sampling in practice.

Chapter 5

Call Admission and Interactive Control for Video Aggregation

5.1 Call admission issues

In the previous chapter, we have discussed different bit-rate allocation methods that can be used in LVAS. Here we shall go into details on how to apply these allocation methods in practice.

In a LVAS video distribution system, transmission is initiated by the clients sending the requested video indexes and their available buffer size to the server. The video server then looks up the video indexes and gets their bit-rate profiles. They are then fed into the aggregation engine, which determines whether, starting within a pre-defined period, the bandwidth is enough to transmit this set of videos. If the answer is yes, these calls are accepted and the videos will be transmitted to the clients right away. Otherwise some requests are dropped in a random manner before they are re-submitted to the aggregation engine. This

process continues until the available bandwidth can support the remaining video requests.

In dropping calls, we can also set priorities among different calls so that calls with a higher Quality of Service (QoS) guarantee will give preference when bandwidth is not sufficient. Those requests that have been dropped may be re-submitted after a period of time. Once a call is accepted, its sending profile is determined. Each sequence can then be transmitted independently according to its own sending profile, without coordination with others, after call admission.

If a new connection request arrives during the transmission of other sequences, the server can take either of the following two strategies. The first is to utilize the unused bit-rate, i.e., the channel rate minus the sum of sending rate for all clients under transmission, and try to see whether it can fulfill the incoming video request. The operations is similar to those discussed in section 2, except that this time the channel bit-rate is not a constant. The second approach is to jointly aggregate both the new and existing calls. If the aggregation is successful, all the sequences are transmitted according to the newly defined schedule. Otherwise, the incoming call would be dropped. Like the situation in the beginning of the transmission, it can re-submit after a pre-defined period of time.

The first approach has the advantage that the time of aggregation is only proportional to the number of incoming calls, and is independent of the number of calls that have already been accepted. However, as the buffer states of the accepted calls will not be altered, the aggregation efficiency is comparatively smaller. The latter approach can achieve greater sharing. Nevertheless, the complexity is larger since it also takes the already-accepted calls into account.

When using the second approach, the bit-rate profiles of the already-accepted call should be adjusted before feeding into the aggregation engine: Time slots before the expected starting time for the new calls should not be considered during this "re-aggregation" process. Furthermore, the buffer states should be consistent in the transition from original transmission schedule to a newer one. Figure 5.1 illustrates these points for an already-accepted sequence when admitting a new call. If the buffer occupancy at the beginning of the new transmission scheme is smaller than that at the end of the original scheme (like the situation shown in the figure), some data are flushed in the buffer and therefore bandwidth is wasted. If the buffer occupancy at the end of the original scheme is larger, it is unrealizable in practice.

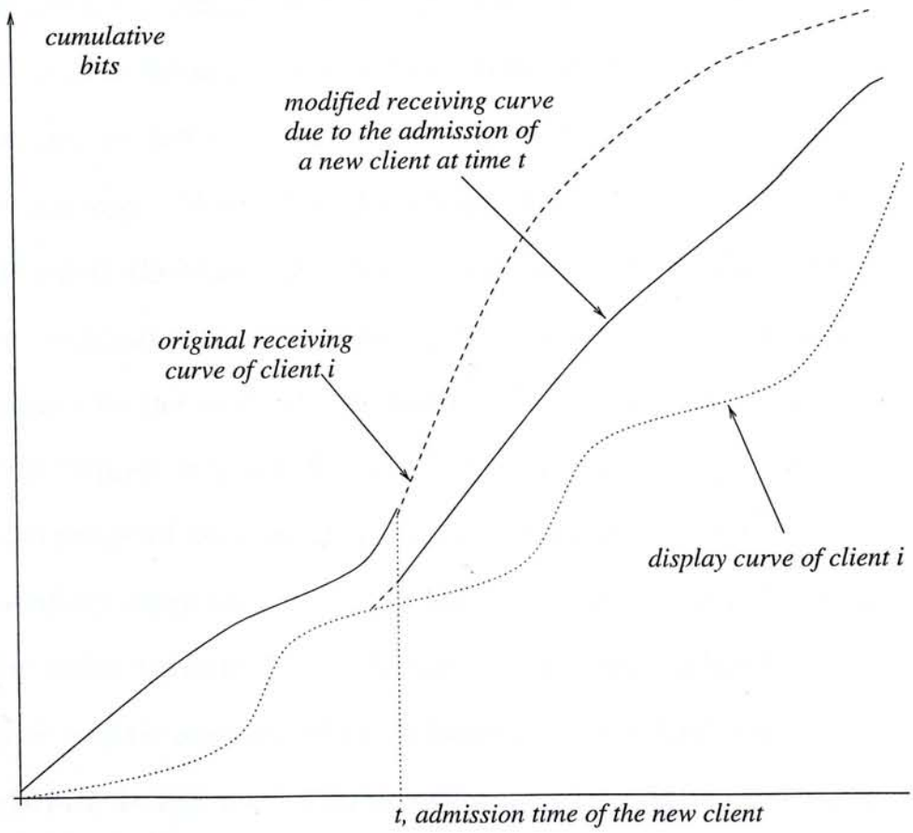

Figure 5.1: The buffer occupancy of the client needs to be consistent

Both approaches would not cause underflow in those already-accepted clients by accepting new sequences.

5.2 Interactive Control

The lossless aggregation system discussed so far is in the form of partial Video-On-Demand service, or so-called 'Pay-per-View' video transmission service. A user can choose which video he wants to enjoy. However, he is not expected to pause, rewind or fast-forward the video. It is because the aggregation scheduler takes advantage of the pre-determined bit-rate profiles of the sequences in order to accept the calls. If the number of bits to be sent in a time slot changes, the video are not guaranteed to be underflow-free anymore.

To enable the users to have interactive control over the video, such as pausing, rewinding or fast-forwarding, the aggregation process can be changed in the following way. Whenever the server receives a user's control command that would affect the transmission schedule, the server performs the aggregation once again, assuming this particular user's session is terminated and re-admitted according to the new bit-rate profile. If underflow of some streams occurs, the control request is ignored and all streams go on playing without affected. The disadvantage of such strategy is that sometimes the users may not be able to successfully issue their controls, because of insufficient bandwidth.

In order to reduce the chance of rejecting interactive commands, we can reserve a little amount of extra bandwidth for the interactivity. The reserved bandwidth is not used during call admission. It is only taken into consideration when users issue interactive controlling commands which would affect the

transmission schedule. Since the behaviour of users in general is unpredictable, it is hard to find precisely how much bandwidth is appropriate for reservation in order to preventing underflow. There are two approaches to this problem. One is to prepare for the worst case. Although it can guarantee the successful issue of control commands, it may suffer from some degradation of bandwidth efficiency due to the somewhat exaggerated assumption. Another approach is to simulate a large number of scenarios to estimate the requirement of reservation. In the simulations below, we take the latter approach to investigate the relationship between the reservation level and the success rate of the interactive control.

5.3 CBR and ABR hybrid

In ATM networks, the above strategy can be transformed to reservation of CBR channel in call admission and using ABR (available bit rate) channel for additional bandwidth requirement in response to the interactive control requests. Since ABR bandwidth is shared with other sessions in the network, the multiplexing gain for the whole network is large and the tariff required will, in turn, be smaller. However, there is an associated problem: as ABR traffic is a best-effort transmission scheme, it is hard to tell exactly how much bandwidth available for a given time slot in future [1]. Therefore we may take some risk that buffer underflow would occur if we use ABR bandwidth in this manner. This will not be the case if we set aside CBR bandwidth for interactivity.

[1] Actually we can guarantee the minimum cell rate (MCR). However, it is supposed to be rather small in order to provide high multiplexing gain.

5.4 Simulation results

An experiment is conducted to study the effectiveness of bandwidth reservation mechanism. For different sets of buffer size and bandwidth combinations, we first find the number of streams that can be allocated using different levels of reservation. After that we randomly shuffle the starting frames of each stream. The effect of shuffling is similar to that of interactive controls in real life, such as fast-forwarding and rewinding. It causes changes in the relative positions of the streams. When a sequence comes to its last frame, it will wrap around to the first one until all frames in the sequences have been played once. We use the shuffled streams to re-do the aggregation. The probabilities that they can be successfully aggregated using all the bandwidth (nominal rate + reserved rate) is recorded. The *frame-equalization* aggregation scheme is used. Let's take an example to illustrate the procedures. For buffer size = 1Mbits and channel rate = 500kbits per second, if we plan to reserve 20% of the bandwidth for interactive control, we first find the number of sequences that can be aggregated using $500k \times (1 - 20\%) = 400$kbps. Let's call this number be a. Then we randomly shuffle the starting time of these a sequences and try re-aggregating them using $500kbps$. We do the shuffling 100 times and record the success rate of re-aggregations after every shuffle.

Note that this experiment is simulating a worst-case scenario: in real life the possibility that all users issue the interactive commands at the same time is small. Therefore the probability where all the bit-rate profiles change is also small. Furthermore, if a user issues a fast-forwarding command in the real situation, the sequence would finish earlier. Thus the demand of bandwidth will

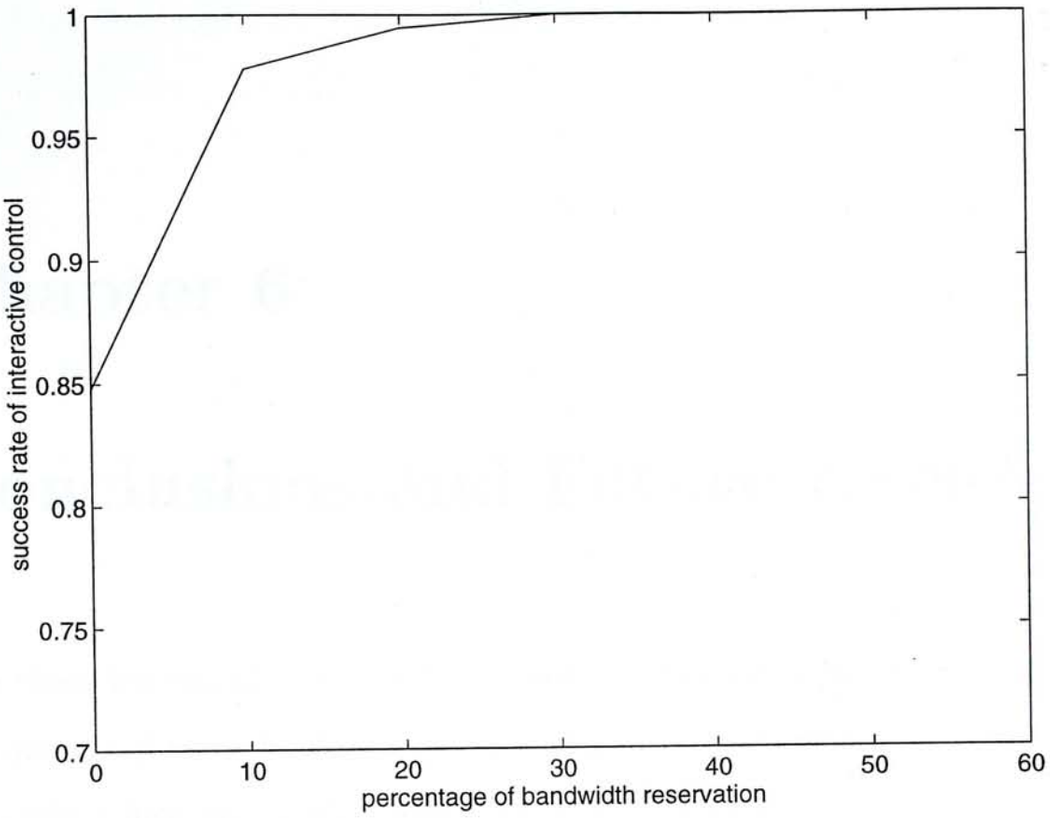

Figure 5.2: Effect of bandwidth reservation to the success rate of interactive control

be smaller as more and more sequences come to the end.

Figure 5.2 are the simulation results of the above experiments. This graph gives us an estimate of the reservation level required in order to achieve a certain success rate of user control. We can see that if we do not reserve bandwidth, the success rate is around 0.85. It becomes higher when the reservation level increases. When the reservation level goes to 30%, almost all the user commands will be accepted.

Chapter 6

Conclusions and Future research

This thesis has considered a *lossless aggregation* framework for the multiplexing of a number of variable-bit-rate videos over a shared communication channel. It can achieve high multiplexing gain without sacrificing video quality.
Four major results and contributions of this thesis are summarized as follows:

- It has been demonstrated that the buffer requirement for the receivers can be significantly reduced using the proposed *lossless aggregation* technique.
- A number of aggregation strategies that can be directly used under the aggregation framework have been investigated. We have identified the *frame-equalization* method to be the most promising technique because it is easy to implement and it has the most outstanding performance in terms of bandwidth efficiency and computation speed.
- A point sampling method has been proposed to further speed up the computation during the aggregation process. The cost of this complexity reduction is the need for a slightly larger receiver buffer and startup delay.

For a fixed buffer size, the point-sampling method generally yields a smaller multiplexing gain. However, if the buffer size is not too small, the degradation in multiplexing gain is also very small. Therefore, we have proved the justifiability of the point sampling technique in practice because the penalty is minimal.

- Call admission and interactive controls in *Lossless Video Aggregation System* have been considered. Based on a large number of simulation experiments, 100% success rate of user interactive control commands (i.e., none of the simulated commands is blocked) can be achieved by reserving about 30% extra bandwidth, compared with the setting without interactive commands.

In conclusion, not only have the advantages of *lossless aggregation* been proved, its feasibility and practicality have been demonstrated in this work.

6.1 Future Research Suggestions

Combination of Lossless and Lossy Adaptations

This thesis is about *lossless aggregation.* We take advantage of the predetermined bit-rate profiles to schedule video transmission. In the last chapter, we saw that the user interactive control can disturb the bit-rate profiles of the on-going videos. To tackle this problem, we can put some *lossy* ingredients into this aggregation framework.

To do so, at the encoding stage, the data in each video stream are grouped into two priorities: a guaranteed stream and an enhancement stream. During call

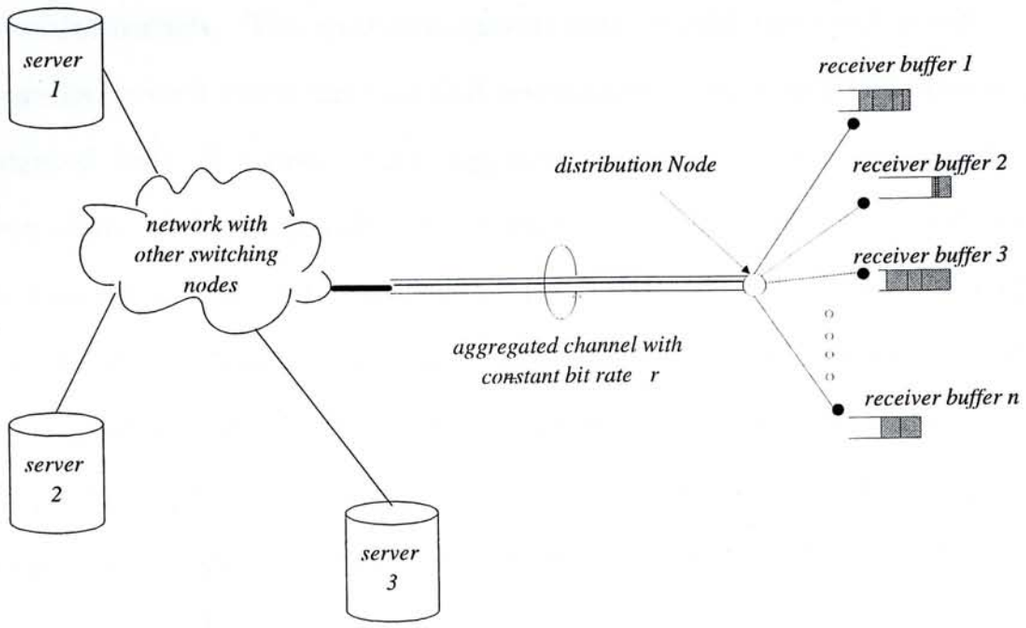

Figure 6.1: A scenario where tight synchronization among servers is required

admission, the full bit-rate profiles (i.e., guaranteed plus enhancement streams) of sequences are considered. After the transmission begins, however, only the guaranteed stream will be used in the aggregation process if a user issues an interactive control that alters the bit-rate profile such that the video stream is no longer supportable by the channel bandwidth. For those clients that receive only their guaranteed streams, the aggregation server will continuously check the available bandwidth to see whether their enhancement streams can also be sent in the future. They will be allowed to send the full-version of the video streams if there is enough bandwidth. This strategy can reduce the blocking probabilities of the user control commands.

Issue of synchronization

If the video streams from a number of distributed servers are aggregated over a common communication channel(see Figure 6.1), there is the new issue

of synchronization. The synchronization issue would not exist if every server aggregates its own video streams and reserves its portion of CBR bit-rate in the aggregated link. However, when aggregation extends to the case where many servers share the aggregated channel (see Figure 6.1), it may cause problems if the servers are not synchronized or the delay jitters are large. Specifically, the servers do not know exactly how much bandwidth are remaining in the link at a given time slot because of delay jitters. This problem can be solved by putting some buffer at the head-end of the CBR channel in order to smooth the delay jitters, or reserving some bandwidth in the link to allow the disalignment between sequences.

Aggregation of distributed servers, however, is worthwhile only if each server is responsible for only a very small number of video streams. As shown in this thesis, aggregating beyond around 15 video streams only brings about marginal gain in performance. Therefore, it is not worthwhile to consider synchronizing distributed servers if each server already has a large number of video to send.

6.2 Publications

Two papers reporting the work partially have been published:

- "Lossless Aggregation for Transporting Stored Video over a CBR Communications Channel", Hanford H. Chan and Soung C. Liew, *Proceedings of International Conference on Image Processing 96.*

- "Lossless Aggregation: A Scheme for Transmitting multiple stored VBR Video Streams over a Shared Communications Channel without Loss of

Image Quality", Soung C. Liew and Hanford H. Chan, *to appear in the August 1997 issue of IEEE Journal on Selected Area of Communication*

Bibliography

[1] Soung C. Liew and C. Y. Tse, "Video aggregation: Adapting video traffic for transport over broadband networks by integrating data compression and statistical multiplexing," *IEEE Journal on selected areas in Communications Vol.14, Issue 6, pp.1123-1137* (short version appeared in *Proceedings of Infocom 95, pp.439-446*).

[2] Soung C. Liew and Chi-Yin Tse, "A control-theoretic framework for adaptation of VBR compressed video for transport over a CBR communication channel" submitted to *IEEE Transactions of Networking.*

[3] ISO/IEC 11172, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbps", 1993

[4] Gertjan Keesman and David Elias "Analysis of joint bit-rate control in multi-program image coding", *SPIE Proceedings Vol. 2308, Visual Communications and Image Processing 94, pp.1906-1917*

[5] Cheng-Tie Chen and Andria Wong, " A self-governing rate buffer control strategy for pseudoconstant bit rate video coding", *IEEE Transactions on Image Processing, January 1993, pp.50-59.*

[6] Masahisa Kawashima, Cheng-Tie Chen, Fure-Ching Jeng and Sharad Singhai "Adaptation of the MPEG video-coding algorithm to Network Applications", *IEEE Transactions on Circuits and Systems for Video Technology, Vol.3 No.4 August 1993, pp.261–269*

[7] D. Reiginger, et al. "Statistical multiplexing of VBR MPEG compressed video on ATM networks", *Proceedings of Infocom 93, Vol.3 pp.919-926*

[8] Dexter Kozen, Yaron Minsky and Brian Smith " Efficient Algorithm for optimal video transmission", *Networked computer science technical report library, http://cs-tr.cs.cornell.edu*

[9] Jean M. McManus and Keith W. Ross, "Video on Demand over ATM: Constant-Rate Transmission and Transport", *Proceedings of Infocom 96, pp.1357-1362*

[10] Juan Miguel Del Rosario and Geoffrey Fox, "Constant Bit Rate Network Transmission of Variable Bit Rate Continuous Media in Video-On-Demand Server", *Multimedia Tools and Applications, Vol. 2, 1996, pp.215-232*

[11] Wu-Chi Feng and Stuart Sechrest, "Critical Bandwidth Allocation for the delivery of compressed video" *Computer Communication Vol. 18 No.10, October 1995*

[12] M. W. Garrett, "Contributions Toward Real-Time Services on Packet Networks",*Ph.D. Dissertation, Columbia University, May 1993*

[13] E. Knightly "H-Bind: A new approach to providing statistical performance guarantees to VBR traffic", *Proceedings of Infocom 96, March 1996*

[14] D. Wrege, E. Knightly, J. Liebeherr and H. Zhang "Deterministic delay bounds for VBR video in packet-switching networks: Fundamental limits and practical trade-offs", *IEEE/ACM Transactions of Networking, Jun 1996, pp.352-362*

[15] E. Knightly and H. Zhang, "Traffic characterization and switch utilization using deterministic bounding interval dependent traffic models", *Proceedings of Infocom 95, Vol.3 pp.1137-1145*

[16] H. Zhang and E. Knightly, "RED-VBR: A renegotiation-based approach to support delay-sensitive VBR video", *ACM Multimedia Systems Journal 5(3), May 1997*

[17] E. Knightly and H. Zhang, "Connection Admission Control for RED-VBR, a renegotiation-based service", *Proceedings of the 4th International IFIP workshop on QoS, March 1996*

[18] H.Zhang and E. Knightly, "A new approach to support delay-sensitive VBR video in packet-switched networks" *Proceedings of NOSSDAV 95, April 1995*

[19] E. Knightly and P. Rossaro, "Effects of smoothing on end-to-end performance guarantees for VBR video", *Proceedings of International Symposium on Multimedia Communications and Video Coding, October 1995*

[20] John Lauderdale and Danny H. K. Tsang "Using the minimum reservation rate for transmission of pre-encoded MPEG VBR videos using CBR service", *IEICE Transactions of Communications, August 1996, pp.1023-1029*

[21] John Lauderdale and Danny H. K. Tsang "An improved algorithm for lossless smoothing of MPEG video with delay constraints", *Proceedings of the International Conference on Visual Information Systems 1996*

[22] Amy R. Reibman and Arthur W. Berger, "Traffic descriptors for VBR video Teleconferencing over ATM networks", *IEE/ACM Transactions on Networking, Vol.3 No.3 June 1995, pp.329-339*

[23] Amy R. Reibman, "Constraints on variable bit-rate video for ATM networks", *IEEE Transactions of Circuits and Systems on Video Technology, Vol. 2 no.4 December 1992, pp.361-372*

[24] D. Raychauduri, D. Reininger, R. Siracusa, "Video transport in ATM networks: a systems view", *Multimedia Systems (1996) 4: pp.305-315*

[25] Raj Jain, "Congestion control and traffic management in ATM networks: Recent advances and a survey", *Computer networks and ISN systems 28 (1996) pp.1723-1738*

[26] Marwan Krunz and Satish Tripathi, "Impact of video scheduling on bandwidth allocation for multiplexed MPEG streams", *ACM Multimedia Systems Journal, 1995*

[27] Marwan Krunz and Satish K. Tripathi, "Statistical characteristics and multiplexing of MPEG streams" *Proceedings of Infocom 95, pp455-462*

[28] Maher Hamdi and James W. Roberts, "Burstiness bound based multiplexing schemes VBR video connections in the B-ISDN", *Proceedings of International Zurich Seminar on Digital Communications, Feb 1996*

[29] Maher Hamdi and Pierre Rolin, "Resource requirements for VBR MPEG traffic in interactive applications", *Broadband communications: networks, services, applications, future directions, Springer Verlag, (ISBN: 3540608958) pp.53-64*

[30] M. Grosslauser, S. Keshav and D. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic"

[31] Sanghoon Lee, Seong Hwan Jang and Jeong Su Lee, "Dynamic bandwidth allocation for multiple VBR video sources", *Proceedings of ICIP 94, Vol.1, pp.268-272*

[32] Simon S. Lam, Simon Chow and David K. Y. Yau, "A lossless smoothing algorithm for compressed video", *IEEE/ACM Transactions on Networking, Vol.4, Issue 5, pp.697-708*

[33] Pramod Pancha and Magda El Zarki, "Bandwidth requirements of variable bit rate MPEG sources in ATM network", *Proceedings of Infocom 93, Vol.3, pp.902-909*

[34] Briand deCleene, Pramod Pancha, Magda El Zarki and Henrik Sorensen, "Comparison of priority partition methods for VBR MPEG", *Proceedings of Infocom 94, Vol.2, pp.689-696*

[35] Tomoaki TANAKA, Sakae OKUBO, Hideo HASHIMOTO and Hiroshi YASUDA, "A study on comparison between VBR and CBR video service in ATM environment", *Proceedings of ICC 92, Vol.1, pp.551-555*

[36] Dallas E. Wrege and Jörg Liebeherr, "Video traffic characterization for multimedia networks with a deterministic service", *Proceedings of Infocom 96, Vol.2, pp.537-544*

[37] Young-Chon Kim, Pal-Jin Lee, Dae-Kyu Choi and Byung-Ok Kim, "Dynamic bandwidth allocation for VBR video sources in ATM based BISDN", *Proceedings of ICAPP 95, Vol.1, pp.103-111*

[38] Keith Hung-Kei Chow and Ming L. Liou, "Simple call admission control and buffer management scheme for multiclass video-on-demand service", *Proceedings of ISCAS 95, Vol.1, pp.466-469*

[39] Domenico Ferrari and Dinesh C. Verma, "A scheme for real-time channel establishment in wide-area networks", *IEEE Journal on selected areas in communications, Vol.8 No.3 April 1990, pp.368 –379*

[40] Wei Ding and Bede Liu, "Rate control of MPEG video coding and recording by Rate-Quantization Modeling", *IEEE Transactions on Circuits and Systems for Video Technology, Vol.6 No.1 February 1996, pp 12–20*

[41] Teresa Andrade and A. Pimenta Alves, "Transmission of MPEG2 applications over ATM network", *Multimedia Transport & Teleservices, Proceedings of International COST 237 Workshop, pp.310-321*

[42] Qin Zheng, Kang G. Shin and Emmanuel Abram-Profeta, "Transmission of compressed motion video over computer networks", *Proceedings of COMPCON Spring 93, pp.37-46*

[43] Richard C. Lau, Paul E. Fleischer and Shaw Min Lei, " Receiver buffer control for variable bit-rate real-tim video", *Proceedings of ICC 92, Vol.1, pp.544-550*

[44] T.D.C. Little and A. Ghafoor, "Scheduling of bandwidth-constrained multimedia traffic", *Computer Communications, Vol.15 No.5, July / August 1992, pp.381-387*