Lossless Aggregation: A Scheme for Transmitting Multiple Stored VBR Video Streams over a Shared Communications Channel Without Loss of Image Quality

Soung C. Liew, *Senior Member, IEEE,* and Hanford H. Chan

***Abstract*—This paper introduces a new concept called *lossless aggregation* for the transmission of video information. It is a scheme for the delivery of variable bit-rate (VBR) video streams from a video server to a group of users over a shared channel. No data are dropped at the source during the adaptation process that reshapes the VBR video traffic to conform to the channel bit-rate characteristics. The transmission schedules of individual video streams evolve in a dynamic way that depends on their relative traffic characteristics. Receiver buffer underflow and overflow are prevented. Therefore, the data delivery process does not cause any loss of image quality. We show that very significant receiver-buffer reduction can be achieved with aggregation compared with the independent transmission of individual video streams over separate channels. Several bandwidth allocation methods for aggregation are studied extensively. The *frame equalization* algorithm stands out in terms of its simplicity and optimality.**

***Index Terms*—Bandwidth allocation, bandwidth compression, call admission, lossless aggregation, scheduling, stored video, transmission.**

I. Introduction

VIDEO information can be transmitted using a variable bit-rate (VBR) or a constant bit-rate (CBR) virtual channel in a broad-band network (e.g., an ATM network). 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. This paper concerns the delivery of stored video over a CBR channel.

A common CBR-transmission strategy that has been extensively studied [1]–[6] 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 produce more data than 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 bit rate. Because of the possible loss of image quality, this is sometimes referred to as *lossy* adaptation.

Manuscript received April 10, 1996; revised September 16, 1996. This work was supported by the RGC Earmarked Grant of the Hong Kong University and Polytechnic Grant Council (Grant Ref: CUHK 477/95E). This paper was presented in part at IEEE ICIP'96, Switzerland.

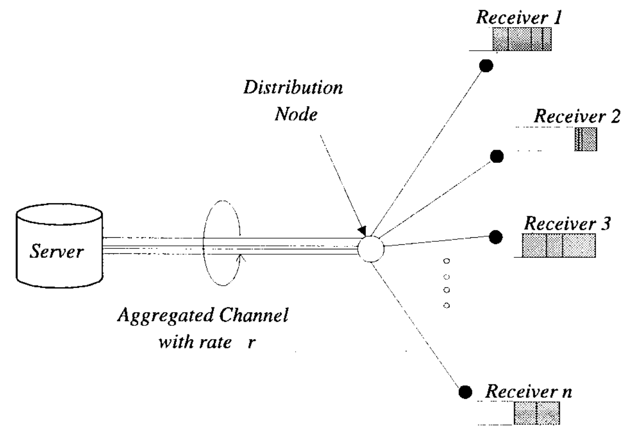
The authors are with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

Publisher Item Identifier S 0733-8716(97)04186-3.

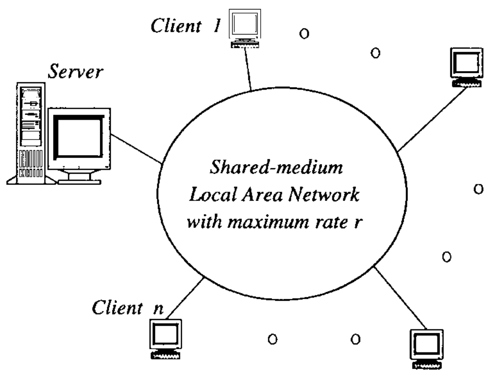
References [7] and [8] proposed a *lossless adaptation* scheme for stored video such as movies. The video is pre-encoded at a constant quality. This means that the data are VBR and would vary over time according to the scene complexity. The main concept in [7] and [8] is to presend a large amount (but not all) of video data to the receiver before the actual display time. A CBR channel is used to deliver the data at a constant rate in such a way that the data will arrive at the receiver before their display time. A concern with this approach, however, is that the receiver buffer for data storage must be very large (e.g., several tens of megabytes) [7].

In this paper, we introduce the concept of *lossless aggregation*, and describe the bit-allocation problem associated with it. This scheme can be used to reduce the receiver buffer requirement in a common networking scenario: a video server serving many clients. As shown in Fig. 1(a), the video server sends a number of video streams to a group of clients over a common CBR channel. Each of the video streams can be VBR, but as a whole, the aggregate bit rate is CBR. A *lossless aggregation* process is used to divide the fixed CBR rate among the video streams in a dynamic fashion. At a remote distribution node in the proximity of the clients, the video streams are separated and sent to their respective clients. We show that the buffer requirements of the receivers can be significantly reduced using lossless aggregation as compared to allocating separate CBR channels to separate video streams.

The remainder of this paper is organized as follows. For motivations, Section II describes some examples of network scenarios in which aggregation can be applied. Section III reviews the lossless transmission of single video over a CBR channel, and presents the basic concept of aggregation. The problem being tackled in Section III is that of minimizing the receiver-buffer size: *How should the CBR channel rate r be set, and how should the transmission of a group of video be scheduled so that the maximum receiver-buffer occupancy among all receivers is minimized?* Based on the framework in Section III, Section IV presents experimental results that demonstrate the effectiveness of aggregation for reducing the receiver-buffer requirements. Sections V and those following consider aggregation in a practical setting: *Given a CBR channel of bit rate r and a fixed receiver-buffer size of B at all the receivers, how should the transmission of a group of video*



(a)



(b)

Fig. 1. Transmission of a group of video streams over (a) a public communication network and (b) a local-area network.

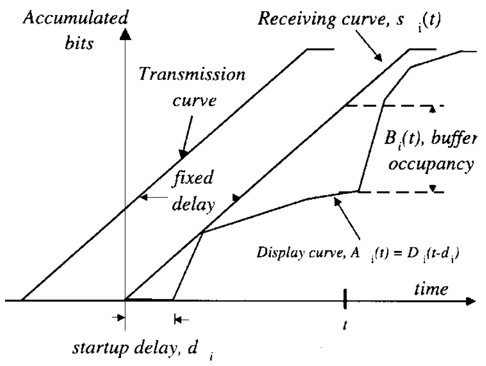


Fig. 2. Graph for illustrating the transmission of a video stream over a CBR channel.

be scheduled so that receiver underflow and overflow would not occur in any of the receivers? Several possible schemes are described in Section VI, and the related experimental results are given in Section VII. Section VIII concludes this paper.

II. Network Scenarios for Application of Aggregation

There are many video-delivery systems in which aggregation 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 two examples in Fig. 1.

Fig. 1(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 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 in this paper.

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 rate of a VC between the distribution node and a client is 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.

Fig. 1(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, then 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. One can also envision a hierarchical video delivery system with a wide-area network (WAN) and LAN's. The server is connected to the WAN, and it sends video streams to gateways that are connected to the WAN as well as LAN's. Through the gateways, the video streams are distributed to individual clients over the LAN's.

Yet another scenario is that of satellite transmission. A video server may send video streams over a CBR uplink to a group of clients at different locations.

III. Conceptual Framework of Aggregation

A. 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 [7]. With reference to Fig. 2, 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 $A_i(t)$ is dictated by the intrinsic bit-rate 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 paper, 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 are stored in a buffer at the receiver. As shown in Fig. 2, the difference between the arrival and display curves in the vertical direction is the amount of data in the buffer at time t.

Strictly speaking, Fig. 2 is only a macroscopic picture of the underlying discrete-time processes described mathematically as follows. The cumulative bits of a video stream i are

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

where $X_i(j)$ = size of the jth frame in sequence i and N_i = number of frames in sequence i. The display curve with start-up delay d_i is then

$$A_i(t, d_i) \triangleq D_i(t - d_i). \tag{2}$$

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

$$s_i(t) = \begin{cases} rt, & t < t' \\ rt', & t \ge t' \end{cases} \tag{3}$$

where $t' = \lceil (D_i(N_i - 1))/r) \rceil$ is the time by which all data in the video have been received.

To prevent buffer underflow (i.e., the situation in which the data to be displayed have 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 start-up delay d_i at time t is

$$B_i(t) \triangleq s_i(t) - A_i(t, d_i) \ge 0, \qquad \forall t. \tag{4}$$

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 touch each other; if not, r can be reduced further. More precisely, the optimal r is given by

$$r = \max_{0 \le t \le d_i + N_i - 1} \left\lceil \frac{A_i(t, d_i)}{t} \right\rceil. \tag{5}$$

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, the reader can check intuitively that there is an optimal d_i at which $\max_t B_i(t)$ is minimized. In [7], experiments have been conducted on the movie *Star Wars*. It was found that the buffer needed is about 22.3 Mbytes, or 6% of the total video size, with a build-up delay of 37 s. The large buffer size is a concern in practical implementation, and the straightforward CBR transmission will need to be modified. This paper offers one possible solution.

[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 in this paper is still valid, albeit a small amount of modification is needed.

B. Lossless Aggregation of VBR Streams

It turns out the receiver buffer requirement can be reduced when a number of videos are transmitted together using a CBR channel, as shown in Fig. 1. 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 receive curve. We can write the aggregate display curve of the n video streams as

$$A(t, \vec{d}) = \sum_{i=1}^{n} A_i(t, d_i) \tag{6}$$

where $\vec{d} \triangleq \{d_1, d_2, \cdots, d_n\}$ is the vector of start-up delays at the receivers. Without loss of generality, we assume $N_1 + d_1 \le N_2 + d_2 \le \cdots \le N_n + d_n$. The aggregate receiving curve for a CBR with rate r is

$$S(t) = \begin{cases} rt, & t < t'_n \\ rt'_n, & t \ge t'_n \end{cases} \tag{7}$$

where $t'_n = \lceil (\vec{A}(N_n + d_n - 1), \vec{d}/r) \rceil$ 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 video i such that its receiving curve is above its display curve. There are many ways to do this, and the following presents a two-phase approach.

In the first phase, we solve the global transmission problem. For a given $\vec{d}$, to minimize the total buffer occupancy, the total receiving rate can be set such that the total receiving curve touches the aggregate display curve at one point. Similar to (5), it is equivalent to choosing $r = \max_{0 \le t \le N_n + d_n - 1} \lceil (A(t, d)t, \vec{d})/t) \rceil$.

Once the total 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:* $\Sigma_i \; B_i(t) = B(t, \vec{d}) \; \forall t$.
- *Constraint 2:* The individual receiving curve $s_i(t) = A_i(t, d_i) + B_i(t)$ is nondecreasing $\forall i, t$.

Once $B_i(t)$ has been determined, $s_i(t)$ can be constructed, and the server can then transmit data to individual receivers according to it. 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.

It turns out that the second-phase problem is hard to solve optimally. A heuristic algorithm, however, can be devised to tackle this problem. A simple idea is to try to equalize the occupancy levels of all receiver buffers. This method, however, has a shortcoming because of the VBR nature of video. Consider, for example, a particular receiver that is about to display a large frame. Suppose that all 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. To overcome this problem, we propose a heuristic that performs buffer-occupancy equalization *backward* in time. The advantage of doing so is that if we see a very large frame at time t, we can use the time slots before t to smooth out the burst.

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

$$B_i(t+1) = B_i(t) - X_i(t-d_i) + \Delta s_i(t) \tag{8}$$

which 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 $s_i(t)$ such that $B_i(t)$ for all i are approximately equal, hence minimizing the maximum buffer occupancy among the streams. This goal can be written as follows:

$$\min; \max_i; B_i(t) \tag{9}$$

where

$$\begin{aligned} B_i(t) &= B_i(t+1) + X_i(t-d_i) - \Delta s_i(t) \\ \sum_i \Delta s_i(t) &= \begin{cases} r, & t < tn' \\ 0, & t \geq tn'. \end{cases} \end{aligned} \tag{10}$$

To achieve this, the n sequences are first sorted in descending order according to the values of $B_i(t)$, which is calculated using the above formula with $\Delta s_i(t) = 0$. 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.

As a concrete example to the above procedure, suppose that $n = 4, r = 8$, and the initial $B_i(t)$ after sorting is 10, 8, 5, 4 units, respectively. At the first round, two units will given to the first sequence, and the $B_i(t)$ becomes 8, 8, 5, 4. Then three units each are given to both the first and second sequences. The updated $B_i(t)$ is now 5, 5, 5, 4. Because the r bits have been used up, the procedure stops here for this time slot.

IV. Demonstration of Buffer Reduction by Experimental Results

Our experiments have indicated 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 [9] to conduct the experiments. The trace recorded the frame sizes of approximately 2 h of the movie *Star Wars*. This trace was used to generate 16 "artificial" traces for simulation experiments. An artificial trace was constructed by concatenating eight pieces of 15-min segments, each of which was extracted from *Star Wars* with a random starting point. That is, to generate a 15-min segment, we randomly and independently chose a starting point s within the 2-h movie, and the interval between s and s plus 15 min formed a 15-min segment.

Our 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 nonaggregated 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 $n = 1, 2, \cdots, 16$, 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 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 reductions for three trace-selection patterns are shown in Table I. The general trend is that the more streams being aggregated, the larger is the buffer reduction. As shown in the table, a buffer reduction factor of more than 20 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 (see Section III), the optimization problem would be easy to solve, although the fact that $s_i(t)$ could decrease with t is nonphysical. 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 assigning the same amount of buffer occupancy to each stream. That is, $B_i(t, d_i) = B(t, \vec{d})/n$ for all i.

Let $B_{\text{opt}} \triangleq \max_{i,t} B_i(t, d_i)$ be the maximum buffer occupancy in the original optimization problem (i.e., with Constraint 2), and let B'_{opt} be that of the modified problem. It is obvious that $B'_{\text{opt}} \leq B_{\text{opt}}$ since the modified problem has one fewer constraint. Using the heuristic algorithm, we

TABLE I
BUFFER REDUCTION FACTOR AND BUFFER PENALTY OF A HEURISTIC AGGREGATION ALGORITHM

	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 %

can also get a buffer requirement B_{heu}. The percentage of buffer penalty due to the use of the suboptimal heuristic is $(B_{\text{heu}} - B_{\text{opt}})/B_{\text{opt}} \times 100\% \leq (B_{\text{heu}} - B'_{\text{opt}})/B'_{\text{opt}} \times 100\%$. The RHS is an overestimate of the buffer penalty. Based on this estimate, the experimental results (Table I) show that the heuristic algorithm is close to optimal. With seven exceptions out of 48 samples, the buffer penalty is smaller than 0.1%.

V. AGGREGATION WITH FIXED CHANNEL RATE AND RECEIVER-BUFFER SIZE

In the preceding section, we have shown that required receiver-buffer size can be reduced quite significantly using 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 heuristic algorithm in Section III).
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 be fixed *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 and subsequent sections look 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?*

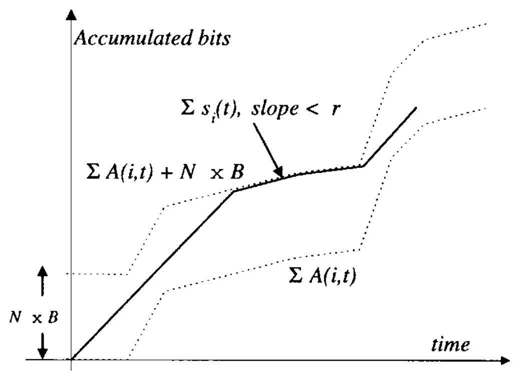


Fig. 3. Aggregation of a bundle of videos. (Note that the aggregate receiving curve $\Sigma_i\ s_i(t)$ is bounded between $\Sigma_i\ A(i,t)$ and $\Sigma_i\ A(i,t) + N \times B$. The slope is bounded above by r.)

Consider the transmission of data from the server to the clients. With reference to Fig. 3, we see that the following global constraints must be satisfied:

$$\sum_{i=1}^{n} A_i(t, d_i) \leq \sum_{i=1}^{n} s_i(t) \leq \sum_{i=1}^{n} A_i(t, d_i) + NxB$$
$$\sum_{i=1}^{n} \Delta s_i(t) = \sum_{i=1}^{n} [s_i(t+1) - s_i(t)] \leq r \quad \forall t. \tag{11}$$

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.

In addition to constraints (11), we also have the following constraints for the prevention of buffer overflow and underflow at individual receivers:

$$A_i(t, d_i) \leq s_i(t) \leq A_i(t, d_i) + B$$
$$s_i(t) \leq s_i(t+1) \qquad \forall t, 1 \leq i \leq n. \tag{12}$$

The first equation of constraint (11) can be derived from that of constraint (12). Hence, there are altogether only three independent inequalities.

The bandwidth-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 $\{s_i(t)$ for all $i, t\}$ 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 infeasibility) very quickly (in a matter of seconds). It turns out that the general problem of determining feasibility/infeasibility in a short time is difficult (as shown in the Appendix). However, 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 such 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 two shortcomings because of 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. The second shortcoming is that the allocation method does not take the future frame characteristics into account.

Section VI considers several (but fast) heuristic scheduling algorithm. The Appendix formulates the general problem of feasibility determination as a dynamic program so as to illustrate the complexity involved. It turns out that some of the heuristics in Section VI have very good performance, hence obviating the need for complicated algorithms.

VI. Bandwidth-Allocation and Scheduling Methods

This section presents and compares several bandwidth-allocation and scheduling schemes. There are two classes of scheduling schemes. The nonaggregation schemes allocate a fixed portion of the channel rate to each stream throughout its duration. The aggregation schemes assign the bits dynamically (i.e., the bits assigned vary from time slot to time slot) to each stream according to its traffic characteristics relative to that of others.

A. Bandwidth Allocation Without Aggregation

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 (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 (r/n). When the buffer is full, 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 update 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.

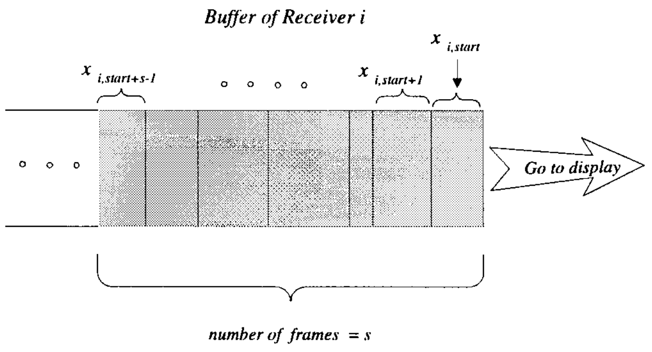


Fig. 4. Notation for describing the buffer states.

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 $r\rho_j/\Sigma_i\ \rho_i$. 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.

Variance Proportional Method: Instead of mean, other statistical indicators such as variances or higher moments can also be used. However, the performances of these fixed bandwidth allocation in general are not as satisfactory as the more dynamic bandwidth-allocation schemes described in the next subsection.

B. Bandwidth Allocation with Aggregation

Partial Look-Ahead Scheduling: The partial look-ahead scheduling scheme is a very general dynamic bandwidth-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). Consider 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 s. Let Δb 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 $start + s - 1$. For each client, we can define an *urgency measure* that describes how urgently the receiver buffer needs more data from the transmitter:

$$\frac{x_{\text{start}+s} - \Delta b}{s}. \tag{13}$$

The motivation for this definition is as follows. In s frame times, the currently partially received frame will need to be displayed at the receiver. This means that the server must transmit $x_{\text{start}+s} - \Delta b$ bits to the client in s 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 $+s+k$ is

$$\frac{\sum_{i=0}^{k} x_{\text{start}+s+i} - \Delta b}{s+k}. \tag{14}$$

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:

$$\max_{0\leq k\leq w-1} \frac{\sum_{i=0}^{k} x_{\text{start}+s+i} - \Delta b}{s+k}. \tag{15}$$

Let U_m be the urgency measure of client m. A bandwidth-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 (15).
- *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

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

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

Let n be the number of streams, and let 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 $O(nl \times nw) = O(n^2 wl)$.

It turns out that, in practice, w need not be very large in order to prevent most of the underflow. In the next section, we show some experimental results which support this statement.

Frame Equalization Method: The frame equalization method is a very simple dynamic bandwidth 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 all the receivers are filled up or no bit rate is left for this time slot. At the end of a time slot, a frame can be partially sent for either of the above reasons. If a frame is partially sent because no bit rate is left for this time slot, the bits of the next time slot will be distributed starting from the remaining part of the frame.

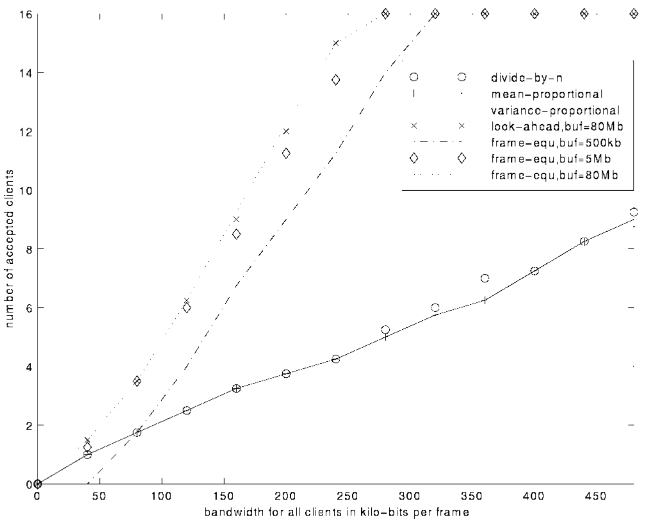


Fig. 5. Average number of sequences accepted using different scheduling methods.

The time complexity is rather small using this method. The total amount 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.

VII. Bandwidth-Allocation Experimental Results

We used the same 16 streams used in Section IV to investigate the performances of different schemes discussed above. 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 bandwidth-allocation methods above were tested. We investigated the number of streams that can be supported with fixed bandwidth r. The start-up delay is ten frames of time. The 16 streams were admitted one by one (according to their 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 this way gives us an indication of how good the bandwidth-allocation method is.

The results of the different bandwidth-allocation schemes are shown in Fig. 5, where the number of accepted sequences is plotted against the channel rate r (in bits/frame). The channel rate ranges from 40 to 480 kbits/frame, spaced 40 kbits/frame apart. Several receiver buffer sizes were tested. The number of sequences in the figure refers to the *average* number of sequences admitted with the four sets of ordered streams.

The effects of receiver buffer size on the number streams admitted are not significant. In fact, the nonaggregation methods are not sensitive to the receiver-buffer size at all: we tested receiver-buffer sizes that range from 5 to 80 Mbits, and they all yielded the same results for the nonaggregation methods. For the aggregation methods, the effect of receiver-buffer size is small, as shown in the figure. This can be explained as follows: when the rate is limited, the receiver buffer size does not help much in the prevention of underflow.

There are very significant differences between the aggregation methods and nonaggregation methods. All the nonaggregation methods are similarly bad, and both of the aggregation methods have roughly the same performance. The number of accepted videos for the nonaggregation methods are significantly lower than that in the aggregation method. For instance, when the bit rate is 240 kbits/frame and the receiver buffer size is 35 Mbits (not shown in the figure), the number of accepted sequences for *partial look ahead* (window size $= 1$) is 14, while that of the *variance proportional method* is only four.

The performance of *frame equalization* is the best among all methods. Together with its small time complexity, it is the best scheme to be used in practice. In a separate experiment, we tried to identify the minimum bit rate r needed so that all 16 streams will be accepted, assuming a receiver-buffer size of 1 Mbyte. A CBR rate of 268 098 bits/frame is needed to admit all 16 streams with frame equalization. The sum total of the mean rates of the 16 streams is is 245 840 bits/frame. Therefore, the bandwidth efficiency is 92%. This is a rather good result, considering the fact that the video streams are being transmitted without loss of image quality.[2]

VIII. Conclusion

This paper has considered lossless adaptation for the transmission of video data over a CBR channel. No data are dropped at the transmitter or the receiver during the adaptation process, and it is easy for the network to guarantee lossless transmission of data over a CBR channel. Therefore, the data-delivery process does not cause any loss of image quality.

Within the lossless adaptation framework, we have introduced and focused on *lossless aggregation*, which is a novel strategy for transmitting a bundle of *stored video* from a server to a group of users over a shared channel. With aggregation, the transmission schedules of the individual video streams are determined dynamically based on their changing relative bit-rate characteristics. Our experimental results have demonstrated that aggregation is a promising technique for significant buffer reduction relative to transporting individual video over separate CBR channels.

Several bandwidth-allocation methods with and without aggregation have also been investigated. It has been shown that given a fixed receiver-buffer size and a fixed channel rate, the aggregation methods can admit more video streams than nonaggregation methods. In particular, the *frame equalization* method outperforms other methods in terms of the number of sequences admitted and the time complexity of the algorithm. With frame equalization, a CBR rate of 268 098 bits/frame is needed to admit all 16 video streams being tested when the buffer size of each receiver is 1 Mbyte and the start-up delay is ten frames of time. Since the sum total of the mean rates of all the 16 streams is 245 840 bits/frame, this gives a bandwidth efficiency of 92%.

There are two ways to use aggregation in practice. In this paper, we have assumed that its use has been integrated into the call admission process. Given its bit-rate characteristics, we examine whether a new video request can be admitted without causing underflow at any of the receivers. If yes, the request will be granted. This prevents underflow altogether. Another way is not to perform such strict call admission. For a channel, we will admit up to a fixed number of video requests. The aggregation algorithms will then be used only for the dynamic scheduling of the video transmission after admission. Underflow may happen, but with a good aggregation algorithm and sufficient channel rate, it should occur rarely. The second mode of operation allows aggregation to be applied in situations in which the channel rate is not CBR. That will be the case, for example, if we use the remaining bit rate to accept incoming calls while some videos are already in the process of aggregation. The same aggregation algorithm can still be used, with the modification that the available bits per time slot may change from slot to slot.

[2]One could perform an experiment in which the video data are packetized into cells at the source, and the VBR traffic of individual video streams is multiplexed within (as opposed to outside) the network onto a CBR channel. To reach a cell-loss rate of zero during the multiplexing process, we expect the bandwidth efficiency to be smaller than that of the aggregation methods. The exact figure remains to be investigated.

Appendix
Dynamic Programming Approach

The general problem of feasibility determination can be formulated as a dynamic program and 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: each time slot corresponds to one stage. A node j at stage t corresponds to a set of values of the individual receiving curves after a time slot t, and it is labeled by a vector indexed by j

$$\begin{aligned}\{\vec{d_j}(t) &\triangleq (d_j(1,t), d_j(2,t), \cdots, d_j(n,t)):\\ &d_j(i,t) \le A(i,t) + r_i, \qquad \forall i, \quad \sum_{i=1}^{n} d_j(i,t) = s(t)\}\end{aligned} \tag{16}$$

where r_i and $A(i,t)$ represent the receiver buffer size and the display curves of the ith video, respectively.

Nodes at adjacent stages are connected by links. The link that connects node $\vec{d_j}(t)$ and node $\vec{d_k}(t+1)$ has a cost function

$$\begin{aligned}&\text{cost}(\vec{d_j}(t), \vec{d_k}(t+1))\\ &\quad = \begin{cases} 1, & \text{if } d_j(i,t) \le d_k(i,t+1) \\ & \quad \le d_j(i,t) + \Delta s(t) \\ & \quad \forall i \in [1, \cdots, n] \\ \infty, & \text{otherwise.}\end{cases}\end{aligned} \tag{17}$$

The above cost function restricts those feasible links to have finite value one. The left part of the inequality condition ensures that the individual receiving curves would not be

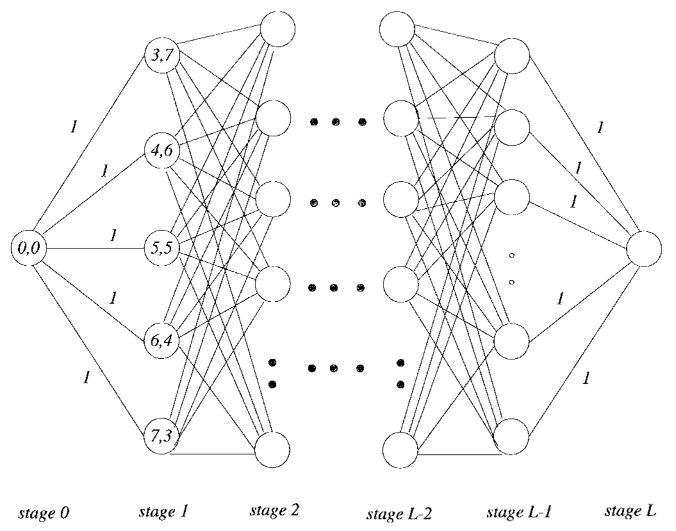


Fig. 6. Typical dynamic-programming network for two streams. The receiver buffer is seven units each, and the start-up delay is greater than one. The first time slot has ten units of bits. Thus, there are five states in the first stage.

decreasing, and the right part of it ensures that the bit rate assigned to any streams would not be larger than the total allocated bit rate for this time slot. If the two conditions are not satisfied, this link will have a cost of ∞. Fig. 6 shows an example that corresponds to the aggregation of two video streams.

Based on this network, the task is to find a path from stage 0 to stage L such that the total cost of it is minimized. (In fact, the minimum value should be L.) This type of program can be solved by conventional *forward dynamic programming*. The *optimal value function* is

$$\begin{aligned} S(\vec{d}_j(t)) &= \text{the minimum cost path from} \\ &\qquad \vec{d}_k(t-1) \text{ to } \vec{d}_j(t) \qquad \forall k \\ &= \min_{\forall k}\{S(\vec{d}_k(t-1)) + \text{cost}(\vec{d}_k(t-1), \vec{d}_j(t))\}. \end{aligned}$$

The *boundary condition* is

$$S(\vec{d}_j(1)) = \text{cost}(\vec{d}_{\{0,0,\cdots,0\}}(0), \vec{d}_j(1)). \tag{18}$$

This model can be solved using the common *dynamic programming* technique. Much simplification can be achieved because those states having infinite value can be neglected in the later calculations. The resulting path(s) found by this method should be optimal in the sense that if it cannot find a path with finite total cost, this set of videos is guaranteed to underflow at some points along the path.

Although the allocation problem can be formulated using *dynamic programming,* in practice, the time complexity is quite high due to the huge number of states. Consider the simple case where each client has a buffer space 5 Mbytes (40 Mbits) which is yet unfilled, and the excessive bit rate for eight clients is 10 Mbits for a particular time slot. There are totally $(40M + 7/7)$ states for this time slot. Such enormous state space prohibits actual implementation. Although we can trade off the complexity with a coarser bit-allocation resolution, the number of states will still be very high for a reasonable scale. Another problem is that there could be many solutions to the dynamic program as far as a feasible schedule is concerned. However, some solutions may be better than others in terms of maximizing the chances of accepting future calls because they make use of the channel rate and receiver buffers more intelligently. This consideration has not been taken into account in the above formulation. These observations motivate the heuristics of Section VI.

REFERENCES

[1] S. C. Liew and C. Y. Tse, "Video aggregation: Adapting video traffic for transport over broadband networks by integrating data compression and statistical multiplexing," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1123–1137, Aug. 1996 (Short version in *Proc. IEEE INFOCOM'95*, pp. 439–446).
[2] ——, "A control-theoretic framework for adaptation of VBR compressed video for transport over a CBR communication channel," *IEEE Trans. Networking*, to appear.
[3] G. Keesman and D. Elias "Analysis of joint bit-rate control in multi-program image coding," in *Proc. SPIE*, vol. 2308.
[4] C.-T. Chen and A. Wong, "A self-governing rate buffer control strategy for pseudoconstant bit rate video coding," *IEEE Trans. Image Processing*, pp. 50–59, Jan. 1993.
[5] D. Reiginger *et al.* "Statistical multiplexing of VBR MPEG compressed video on ATM networks," *Proc. Infocom 93*, vol. 3, pp. 919–926.
[6] D. Kozen, Y. Minsky, and B. Smith, "Efficient algorithm for optimal video transmission," *Networked Comput. Sci. Tech. Rep. Library, http://cs-tr.cs.cornell.edu*.
[7] J. M. McManus and K. W. Ross, "Video on demand over ATM: Constant-rate transmission and transport," in *Proc. IEEE INFOCOM '96*, pp. 1357–1362.
[8] W.-C. Feng and S. Sechrest, "Critical bandwidth allocation for the delivery of compressed video," *Comput. Commun.*, vol. 18, Oct. 1995.
[9] M. W. Garrett, "Contributions toward real-time services on packet networks," Ph.D. dissertation, Columbia Univ., New York, NY, May 1993.

Soung C. Liew (SM'93) received the S.B., S.M., E.E., and Ph.D. degrees in electrical engineering from Massachusetts Institute of Technology, Cambridge, in 1984, 1986, 1986, and 1988, respectively.

He joined Bellcore, NJ, in March 1988. Since July 1993, he has been Associate Professor at The Chinese University of Hong Kong. He has diverse research interests, and has published actively in various areas related to broad-band communications, including fast packet switching, broad-band network control, video transport, and WDM optical networks. His recent research efforts center around the issue of processing and adapting video information for efficient network transport. In November 1995, he launched a project to explore how advanced multimedia and networking technologies can be used to improve university education. The goal is to build a Web-like multimedia lecture-on-demand system. He initiated and is coordinating an ATM testbed network which interconnects three major universities in Hong Kong. He holds two U.S. patents in routing and packet switching.

Dr. Liew is a member of Sigma Xi and Tau Beta Pi.

Hanford H. Chan received the B.Eng. and M.Phil. degrees in information engineering from The Chinese University of Hong Kong (CUHK) in 1995 and 1997, respectively.

He is currently a Research Assistant in the Broadband Communications Laboratory, CUHK. His research interests are in video transmission and application development over broad-band networks.