A Reliable, Adaptive Network Protocol for Video Transport*

Pawan Goyal, **Harrick M. Vin**, **Chia Shen**† and **Prashant J. Shenoy**

Distributed Multimedia Computing Laboratory
Department of Computer Sciences, UT Austin
Taylor Hall 2.124, Austin, TX 78712
Email: {pawang,vin,shenoy}@cs.utexas.edu

† Mitsubishi Electric Research Laboratory
Cambridge Research Center
201 Broadway, Cambridge, MA 02139
Email: shen@merl.com

Abstract

In this paper, we present an adaptive network layer protocol for VBR video transport. It: (1) minimizes buffer requirement in the network while guaranteeing that packets of VBR encoded video flows will not be lost, and (2) minimizes end-to-end delay and jitter of frames. To achieve the former objective, we utilize receiver-oriented adaptive credit-based flow control algorithm, and derive necessary and sufficient number of buffers that should be reserved for ensuring its reliability. To minimize the end-to-end delay and jitter for VBR encoded video streams, we: (1) present bandwidth estimation techniques which exploit the structure of the video traffic, and (2) define a new fairness criteria for buffer allocation and then present a fair buffer/bandwidth allocation algorithm. We experimentally evaluate this protocol for a wide range of parameters and many network configuration and demonstrate its adaptability. We also compare the performance of the protocol with numerous other schemes and demonstrate its suitability for video transport.

1 Introduction

1.1 Motivation

Multimedia applications involving transmission of digital audio and video streams over networks require the network to provide low delay, delay jitter and packet loss. Such applications can be broadly classified as requiring either: (1) mathematically provable, deterministic or statistical quality of service (QoS) guarantees for each video communication channel, or (2) acceptable QoS, but no absolute guarantees. In architectures that support the former class of applications, the network provides QoS guarantees based on the traffic specification of a source [2]. The network guarantees that as long as a source traffic conforms to its specification, it would receive guaranteed QoS. A fundamental limitation of all of these architectures is that most traffic specification models are inadequate for accurately characterizing video traffic. This is mainly because they fail to capture: (1) the short term burstiness resulting from variation in complexity of adjacent frames; and (2) the long term variation resulting from variation in scene complexity. The difficulty in capturing these variations severely limit the effectiveness and viability of architectures for providing guaranteed QoS.

In architectures that support the latter class of applications (referred to as *adaptive* applications), the quality of service being provided to applications varies with the availability of resources [2]. To provide acceptable QoS, a network must manage the burstiness of video traffic at multiple time-scales. The long term bit rate variations lead to sustained overload on a network. Hence, a network must employ an admission control algorithm that limits the statistical multiplexing [4]. The short term burstiness, on the other hand, leads to queue build up, and consequently to significant packet losses, at switches. Although uncompressed video streams, due to the inherent spatial and temporal redundancy, are highly resilient to packet losses, compressed video streams are not. Moreover, if the compression algorithm employs interframe compression techniques (e.g. MPEG), then the effect of packet loss in a single frame may propagate through multiple frames. Consequently, packet losses significantly degrade video quality and are highly undesirable [5].

Given that the hard timing constraints imposed by continuous video playback preclude any retransmission of lost packets, minimizing the impact of such packet losses on video quality will require the sender and the receiver to employ additional error recovery methods. For instance, at a receiver, the decoder may mitigate the effect of lost packets by approximating the missing data through temporal or spatial interpolation [12]. Similarly, the decoder could utilize forward error correction information to compensate for packet losses [1]. Finally, if the network supports multiple priority levels and if the source employs a

*This research was supported in part by IBM Graduate Fellowship, Intel, Mitsubishi Electric Research Laboratories (MERL), the National Science Foundation (Research Initiation Award CCR-9409666), NASA, and Sun Microsystems Inc.

multi-layer encoder, then the reconstructed image quality at the receiver can be improved by transmitting layers of the encoded video stream at different priority levels (e.g., transmitting the essential and the enhancement layers at high and low priorities, respectively) [5, 11]. Although conceptually elegant, these techniques are not without limitations. Whereas the additional data traffic yielded by the redundant forward error correction information increases the overall load and hence may worsen the loss rate; layered encoders, in general, are more complex and require higher bandwidth as compared to a standard single-layer encoder.

Since most of the error recovery techniques increase the complexity of the system, packet losses for compressed video should be avoided. A network can achieve this objective by either: (1) employing source traffic shaping algorithm and removing the short term rate fluctuations, or (2) absorbing the transient overloads by increasing the buffer space at the switches. Source traffic shaping algorithms increase the end-to-end delay of the frames significantly (by approximately 200ms [9]) and hence may not be feasible for interactive video applications. On the other hand, predicting the buffer space required to absorb transient overloads for video sources is difficult. Furthermore, increasing fast expensive buffer at the switches in high speed networks may not be economically viable.

Observe that source traffic shaping algorithms and techniques for transmitting VBR video streams without any shaping are two ends of a spectrum. A network layer protocol for video transport that achieves a middle ground between these two extremes by utilizing hop-by-hop flow control is the subject matter of this paper.

1.2 Relation to Previous Work

Flow control algorithms shape source traffic to avoid congestion in the network. Since these algorithms introduce delay at a source only if necessary to avoid congestion, the delay experienced by flow controlled sources would be smaller than the delay incurred by sources which rely on source only traffic shaping. Hence, flow control algorithms are attractive for video transport. Over the last decade, feedback-based flow control has been the focus of considerable research for datagram networks [5, 6]. However, it has been observed that, due to the end-to-end nature of most of the conventional flow control protocols, they do not react quickly to short-term congestion in high-speed networks. To address this limitation, several researchers have recently begun developing per link, hop-by-hop *credit-based* flow control algorithms [6, 8].

The basic idea in credit based flow control is to reserve buffer for a communication channel (hereafter referred to as *flow*) at all the switches along the path from the source to the destination, and then limit the number of packets an upstream node (hereafter referred to as *sender*) can transmit to a downstream node (hereafter referred to as *receiver*) such that buffer at the downstream node does not overflow. The buffer reserved for a flow depends on the desired bandwidth of a flow and can be either *static* or *dynamic* (i.e., *adaptive*) [6, 8, 10]. In static allocation, a fixed number of packet buffers are allocated at the receiver for each flow. An adaptive credit allocation scheme, on the other hand, permits a number of flows to dynamically share the same buffer pool by adjusting the buffer allocation in accordance with the actual bandwidth usage of each flow. This enables the switching nodes to minimize the total buffer space required to support the flows whose bandwidth requirements vary significantly over time (e.g. video flows).

Adaptive credit allocation can be done either at the sender or at the receiver [6, 10]. In either case, the adaptive credit allocation algorithm must ensure that packets are never dropped due to congestion in the network. A sender based buffer adaptation algorithm was proposed and demonstrated to be reliable in [10]. The concept of receiver-oriented adaptation, on the other hand, was introduced in [6] and a particular algorithm has been proposed in [7].

To ensure that packets are not discarded due to congestion in receiver-oriented adaptation, the algorithm in [7] uses a highly conservative buffer reservation policy, and hence leads to low link and buffer utilization as well as poor delay performance for video flows (see Section 4.1). The algorithm has been designed and evaluated only for data traffic and is not suitable for video traffic (in fact, for video flows the static buffer allocation algorithm requires fewer buffers than the algorithm proposed in [7]). Furthermore, strong assumptions and additional support from senders is required to ensure the reliability of the algorithm [3]. We address these limitations by: (1) deriving a necessary and sufficient condition for ensuring the reliability of receiver-based adaptation, (2) developing a adaptive buffer allocation algorithm specifically for video transport, and (3) demonstrating the viability of employing it for video transport over high speed networks.

The rest of the paper is organized as follows: In Section 2 we present our network protocol for video transport. Section 3 describes the results of our simulations, and Section 4 compares our protocol with other schemes and algorithms. Finally, Section 5 summarizes our results.

2 Network Protocol for Video Transport

Our network protocol for video transport has two main objectives: (1) minimize buffer requirement while ensuring that packet losses do not occur, and (2) minimize end-to-end delay and jitter of frames. In what follows, we first achieve the former objective by deriving necessary and sufficient number of buffers that should be reserved for ensuring the reliability of receiver-oriented adaptive credit-based flow control, and then present techniques for minimizing the end-to-end delay and jitter for VBR encoded video streams.

2.1 Minimizing Buffer and Ensuring Reliability

In credit-based flow control, to ensure that none of the packets transmitted by a sender are discarded due to lack of buffers, the receiver is required to periodically inform the senders about the buffer space availability for each flow. A robust protocol (referred to as *Credit Update Protocol*) for sending credit packets and updating the buffer availability information at the sender was proposed in [6]. The essential elements of this protocol are as follows:

For each flow f, the sender maintains two values: (1) a transmit count T_f, which denotes the number of packets transmitted for flow f, and (2) a credit balance CB_f, which indicates the buffer availability for the flow at the receiver. The receiver, on the other hand, maintains: (1) a forward count F_f, which denotes the number of packets of flow f forwarded by the receiver to a downstream node, and (2) A_f, the number of buffers allocated for flow f. The values of T_f and F_f are initialized to 0, and CB_f is initialized to A_f at the flow setup time. A sender transmits a packet of flow f to a receiver only if $\mathrm{CB}_f > 0$. Whereas the sender increments T_f by 1 and decrements CB_f by 1 on transmitting a packet, the receiver increments F_f by 1 on forwarding a packet to a downstream node. Once the receiver has forwarded a fixed number of packets to a downstream node, it sends a *credit update packet* containing the value $(\mathrm{A}_f + \mathrm{F}_f)$. Upon receiving this packet, the sender updates its credit balance value as: $\mathrm{CB}_f = (\mathrm{A}_f + \mathrm{F}_f) - \mathrm{T}_f = \mathrm{A}_f - (\mathrm{T}_f - \mathrm{F}_f)$. Notice that since $(\mathrm{T}_f - \mathrm{F}_f)$ denotes the total number of packets that are either in transit from the sender to the receiver or are queued at the receiver, $\mathrm{CB}_f = \mathrm{A}_f - (\mathrm{T}_f - \mathrm{F}_f)$ denotes the number of free packet buffers available at the receiver for packets of flow f.

As described in [6], at any instant, the buffer allocation for a flow at the receiver can be partitioned as $\mathrm{A}_f = (\mathrm{N}_2 + \mathrm{N}_3)$. Here, N_2 denotes the number of packets that must be forwarded by a receiver to a downstream node before transmitting a credit update packet to a sender, and thereby determines the bandwidth overhead of transmitting credit update packets. The value of N_3, on the other hand, depends on the bandwidth required by the flow. Specifically, if RTT is the round-trip propagation delay between the sender and the receiver, and if BW (expressed in terms of packets/sec) denotes the bandwidth required by the flow, then N_3 can be defined as $\mathrm{N}_3 = \mathrm{BW} * \mathrm{RTT}$.

Notice that the value of N_2 is a parameter of the credit update protocol, and hence does not vary with time [1]. However, to efficiently share the buffer pool available at the receiver, the value of N_3 assigned to each flow must be dynamically altered so as to match the variations in their bandwidth requirements. Observe that the bandwidth requirement of a flow can either be estimated based on past measurements or explicitly notified by the source. In either case, once the value of N_3 is determined, the receiver-based buffer management algorithm must reserve appropriate amounts of buffer space for each flow, and notify the sender of the allocation. Specifically, if t_f^j and A_f^j denote the time instant for the j^{th} $(j \geq 1)$ buffer reallocation and the corresponding credit allocation for flow f, respectively, then following the reallocation, the receiver must transmit a credit update packet containing the value $(\mathrm{A}_f^j + \mathrm{F}_f)$ to sender of flow f. Due to the non-zero delay involved in communicating buffer reallocation information to the senders, a sender may transmit several packets to the receiver prior to the receipt of the credit update packet. Consequently, the adaptive buffer management algorithm must reserve sufficient number of buffers at the receiver so as to ensure that none of the packets transmitted by the sender during the transition period are discarded due to buffer overflow.

To derive sufficient number of buffers that must be reserved at the receiver for a flow to ensure reliable transmission, we assume: (1) The credit update cell transmitted by the receiver as a result of a reallocation at time t_f^j is guaranteed to be processed by the sender by time $t_f^j + \delta_f$; and (2) If ρ_f denotes the delay between the sender sending a packet and the receiver receiving it, then $t_f^{j+1} - t_f^j > \delta_f + \rho_f \quad j \geq 1$. We refer to $(t_f^{j+1} - t_f^j)$ as adaptation interval. Note that the first assumption states that the sum of queuing, transmission and processing delay for a credit update packet is bounded. The second assumption, on the other hand, limits the frequency of buffer reallocation, and thereby enables reliable bandwidth utilization measurements. We believe that these assumptions are not restrictive and are, in fact, desirable.

Given these assumptions, to formulate the buffer requirement for a flow f at time t, let us first partition the set S of flows sharing a common pool of buffers at the receiver into sets $N(t)$ and $O(t)$, such that $N(t)$ denotes the set of flows for which $(\rho_f + \delta_f)$ units of time have not elapsed since their most recent reallocation and $O(t)$ contains all of the remaining flows. Formally,

$$N(t) = \{i \in S | \exists j \geq 1 \ni (t_f^j \leq t) \wedge (t - t_f^j \leq \delta_f + \rho_f)\} \quad (1)$$

and $O(t) = S - N(t)$. Observe that $O(t)$ consists of the flows for which the new allocation has been processed at

[1] It is desirable to keep N_2 constant for video sources which continuously transmit data.

the sender and all the packets reflecting previous allocation have arrived. Hence, intuitively, the buffer requirement for flows in $O(t)$ at time t, denoted by $B_f(t)$, is $\max\{Q_f(t), \mathrm{A}_f^j\}$ where $Q_f(t)$ is the queue occupancy at the receiver at time t. On the other hand, for the flows in $N(t)$, packets reflecting the previous allocation may not have arrived at the receiver. Hence, for flows in $N(t)$, $B_f(t) = \max\{Q_f(t), \mathrm{A}_f^{j-1}, \mathrm{A}_f^j\}$. Hence, if B is the buffer space available at the receiver, then packet loss would not occur if $\sum_{f \in S} B_f(t) \leq B$ at all time t. This is formally stated in Theorem 1, proof for which is presented in [3].

Theorem 1 *If B denotes the the buffer space available at a receiver, then packet loss would not occur at the receiver if:*

$$\sum_{f \in S} B_f(t) \leq B$$

where $B_f(t) = \max\{Q_f(t), \mathrm{A}_f^{j-1}, \mathrm{A}_f^j\}$ if $f \in N(t)$ and $B_f(t) = \max\{Q_f(t), \mathrm{A}_f^j\}$ if $f \in O(t)$. $Q_f(t)$ is the queue occupancy at the receiver at time t.

Now, consider a buffer allocation algorithm which reserves $B_f(t_f^j)$ amount of buffer for flow f for the interval $[t_f^j, t_f^{j+1})$. To ensure that no packet loss occurs, the actual buffer requirement for flow f must never exceed $B_f(t_f^j)$ at any time instant within the interval $[t_f^j, t_f^{j+1})$ (i.e., $B_f(t) \leq B_f(t_f^j); t \in [t_f^j, t_f^{j+1})$). We have shown that this holds for $B_f(t)$ as defined in Theorem 1 [3].

An important property of $B_f(t)$ is that all buffer requirement functions that can ensure that packet loss does not occur and are non increasing between two consecutive adaptation instants have values at least as large as $B_f(t)$ between consecutive adaptation instants. Hence, any buffer allocation algorithm that reserves sufficient buffer space at adaptation instants (to prevent any buffer overflow within the adaptation interval) must *necessarily* reserve $B_f(t)$ buffer per flow. This is formally stated in Theorem 2, proof for which is presented in [3].

Theorem 2 *$G_f(t) \geq B_f(t)$ where $G_f(t)$ is a buffer requirement function that ensures that no packet loss occurs and is non increasing in the interval $[t_f^j, t_f^{j+1})$ $j \geq 1$.*

Theorem 1 minimizes buffer requirement while eliminating any adverse effects of packet losses on the recovery of compressed images. Techniques for minimizing the end-to-end delay of video frames are presented in the next section.

2.2 Minimizing End-to-End Delay

To minimize end-to-end delay and jitter, our adaptive protocol dynamically allocates bandwidth. To dynamically allocate bandwidth, the network should first estimate the bandwidth requirements of the flows and then allocate buffers corresponding to the bandwidth requirements. Additionally, in the event that the cumulative buffer requirement exceeds the buffer space availability, a buffer allocation algorithm must distribute the buffer space among the competing flows in a *fair* manner (i.e., equitably distribute the increase in end-to-end delay among all the competing flows). In what follows, we present bandwidth estimation techniques and a fair buffer allocation algorithm which achieve these objectives.

2.2.1 Bandwidth Estimation

The bandwidth requirement of a video flow changes with the frame size. Since frames are generated at a constant rate, the bandwidth requirement varies periodically. Hence, a network can measure the bandwidth utilized over an adaptation (measurement) interval and utilize it to estimate the bandwidth requirement during the next adaptation interval, and then allocate the bandwidth accordingly. Specifically, if BW_f^{cur} denotes the bandwidth utilized by flow f during the current adaptation interval, BW_f^{min} and BW_f^{max} denote the minimum and the maximum bandwidth desired by flow f and γ_f ($\gamma_f \geq 1$) the *rampup factor* for flow f, then the bandwidth requirement of the flow during the next adaptation interval can be estimated as:

$$\mathrm{BW}_f^{est} = \min\{\mathrm{BW}_f^{max}, \gamma_f * \max\{\mathrm{BW}_f^{min}, \mathrm{BW}_f^{cur}\}\} \quad (2)$$

The choice of a rampup factor depends on the burstiness of the traffic. For video flows, burstiness can be measured in terms of the ratio of successive frame sizes, which, in turn, depends on the compression algorithm. In fact, if, for a compression algorithm, the maximum ratio of successive frame sizes (denoted by R_f^{max}) is estimated (possibly by analyzing a large number of video streams encoded using the compression algorithm), then choosing $\gamma_f = \mathrm{R}_f^{max}$ will enable a switch to quickly rampup to the maximum bandwidth requirement of flow f whenever desired. However, since not all successive frames of flow f require an increase in bandwidth estimated by the ratio R_f^{max}, such a technique may lead to severe under-utilization of available buffer space. A conservative value of γ_f (i.e., $\gamma_f \ll \mathrm{R}_f^{max}$), on the other hand, yields high buffer space utilization, but at the expense of increased end-to-end delay. Consequently, the rampup factor should be selected such that it balances the end-to-end delay with buffer space utilization.

Observe that a single value of rampup factor is sufficient for intra-frame compression algorithms (e.g., JPEG). However, compression algorithms that exploit inter-frame dependencies (e.g., MPEG) yield different types of frames, each with a different bandwidth requirement. Consequently, selecting the same rampup factor without considering the type of frame being transmitted may yield significant variation in end-to-end frame delays (i.e., high jitter). Hence, for inter-frame compression algorithms, minimizing the delay jitter requires the selection of frame-specific rampup factors. Such frame-specific rampup factors can be provided by the source to the network either: (1) by explicitly transmitting a control packet prior to transmitting each frame, or (2) by providing the information at the time of connection establishment (e.g., by specifying the encoding pattern and the corresponding rampup factors). By *implicitly* specifying the bandwidth requirement of a frame, such frame-specific rampup factors minimize the delay jitter and buffer space requirement at the switches.

Once the bandwidth requirement of a flow has been estimated, a switch allocates buffer corresponding to the estimated bandwidth. To reduce the computational requirement at a switch, we assume that the buffer allocation algorithm allocates buffer for all the flows at the same time. If at the reallocation instant, the cumulative buffer requirement of all the flows is smaller than the available buffer space, then the requirements of all the flows can be met. On the other hand, if the cumulative requirement exceeds the buffer space availability, then the algorithm must achieve a *fair* distribution of the buffer space among all the competing flows. In what follows, we define our fairness criteria and present an algorithm which allocates buffer fairly.

2.2.2 Buffer Allocation Algorithm

The fairness criteria for buffer allocation depends on the requirements of the applications. Since frame delay is a critical QoS parameter for video flows, a buffer allocation algorithm can be considered fair if it uniformly distributes the increase in delay yielded by limited buffer space availability among all the flows. To precisely define the fairness requirement, let us denote the desired bandwidth and the available bandwidth (resulting from the buffer space constraints) for flow f by BW_f and $\widehat{\mathrm{BW}}_f$, respectively. Since the size of a video frame is the same regardless of the the bandwidth allocation, we get:

$$\mathrm{BW}_f * \Delta_f = \widehat{\mathrm{BW}}_f * \widehat{\Delta}_f \Rightarrow \frac{\mathrm{BW}_f}{\widehat{\mathrm{BW}}_f} = \frac{\widehat{\Delta}_f}{\Delta_f} \qquad (3)$$

where Δ_f and $\widehat{\Delta}_f$ denote the delay experienced by a frame of flow f when the bandwidth allocated is BW_f and $\widehat{\mathrm{BW}}_f$, respectively. Then, we define the buffer allocation to be fair if, for all flows in S, the following condition holds [2]:

$$\forall f \in S: \ \frac{\mathrm{BW}_f}{\widehat{\mathrm{BW}}_f} = \frac{\widehat{\Delta}_f}{\Delta_f} = \alpha \qquad (4)$$

[2]For ease of exposition we have assumed uniform fairness. Analysis for weighted fairness can be carried out similarly.

Hence, the main objective of the fair buffer allocation algorithm is to determine α, which in turn determines the buffers that can be allocated for each flow.

For simplicity of presentation, let us assume that j^{th} allocation is being computed for each flow. Moreover, let us assume that the length of the adaptation interval is such that for all flows $t_f^j - t_f^{j-1} > \delta_f + \rho_f$. Hence, if the adaptation occurs at time t, all the flows in S belong to set $O(t)$ (see Equation (1)). Consequently, as per Theorem 1, the minimum buffer requirement of each flow f prior to the reallocation can be given by $(r_f^{j-1} + \mathrm{N}_2)$, where:

$$r_f^{j-1} = \max\{Q_f(t_f^j), \mathrm{A}_f^{j-1}\} - \mathrm{N}_2 \qquad (5)$$

Consequently, the total number of buffer that are available for reallocation is given by:

$$\mathcal{A} = \mathcal{B} - \sum_{f \in S} \left(\mathrm{N}_2 + r_f^{j-1}\right) \qquad (6)$$

where $\mathcal{B}$ denotes the total buffer space available at the receiver.

Now, let b_f denote the buffer allocation for desired bandwidth BW_f for flow f. Depending on the relationship between the b_f and r_f^{j-1}, the set of flows S can be partitioned into two subsets:

$$L = \left\{ f \mid f \in S \ \wedge \ b_f \leq r_f^{j-1} \right\}$$

and $H = S - L$. Since $\forall f \in L$, the desired buffer allocation is smaller than the current allocation, the new allocation can be reduced to:

$$\forall f \in L: \ \mathrm{A}_f^j = (b_f + \mathrm{N}_2)$$

However, since after reallocation the flows belongs to $N(t)$, as per Theorem 1 the buffer that must be reserved for ensuring reliability is given as:

$$\max\{Q_f(t_f^j), \mathrm{A}_f^{j-1}, \mathrm{A}_f^j\} = \max\{Q_f(t_f^j), \mathrm{A}_f^{j-1}\} = r_f^{j-1} + \mathrm{N}_2$$

On the other hand, for all the flows f belonging to H, $b_f > r_f$. Let us denote:

$$b_f = r_f^{j-1} + x_f \qquad (7)$$

In such a scenario, whether or not each flow in H receives its desired bandwidth depends on the current buffer space availability at the receiver. Specifically, if $\sum_{f \in H} x_f \leq \mathcal{A}$, then $\forall f \in H: \ \mathrm{A}_f^j = r_f^j = (b_f + \mathrm{N}_2)$. On the other hand, if $\sum_{f \in H} x_f > \mathcal{A}$, then the number of buffers that can be allocated to each flow f is likely to be smaller than b_f. To precisely compute the allocations, let us denote the bandwidth and the corresponding buffers that can be allocated to flow f by $\widehat{\text{BW}}_f$ and $\widehat{b}_f$, respectively. Let $\widehat{b}_f$ be divided as $(r_f^{j-1} + y_f)$. Clearly, since $\widehat{b}_f \le b_f$, we get $y_f \le x_f$. Moreover,

$$\sum_{f \in H} y_f = \mathcal{A} \tag{8}$$

Now, as per the fairness criteria (see Equation (4)), for all flows $f \in H$, we need to determine y_f such that:

$$\forall f \in S: \ \frac{\text{BW}_f}{\widehat{\text{BW}}_f} = \frac{\frac{b_f}{\text{RTT}_f}}{\frac{\widehat{b}_f}{\text{RTT}_f}} = \alpha \tag{9}$$

Substituting the values of b_f and $\widehat{b}_f$, we get:

$$\frac{r_f^{j-1} + x_f}{r_f^{j-1} + y_f} = \alpha \Rightarrow y_f = \frac{r_f^{j-1} + x_f}{\alpha} - r_f^{j-1} \tag{10}$$

Hence, from Equations (10) and (8), we get:

$$\alpha = \frac{\sum_{f \in H}(r_f^{j-1} + x_f)}{\sum_{f \in H} r_f^{j-1} + \mathcal{A}} \tag{11}$$

Hence, for each flow f, the value of y_f can be derived as:

$$y_f = \frac{\mathcal{A} * (r_f^{j-1} + x_f) + x_f * \sum_{i \in H} r_i^{j-1} - r_f^{j-1} * \sum_{i \in H} x_i}{\sum_{i \in H}(r_i^{j-1} + x_i)} \tag{12}$$

An interesting point to note is that Equation (12) does not guarantee that y_f will exceed zero for all $f \in H$. That is, to achieve fair distribution of buffer space as per our fairness criteria (see Equation (4)), a subset of the flows in set H may be required to release some of the buffers that have been reserved for them during the interval $[t_f^{j-1}, t_f^j)$. The occurrence of this condition (namely, $y_f < 0$) is indicative of the fact that some of the flows could have been allocated unfair share of the total buffer space during the previous adaptation interval.

Notice, however, that to ensure reliability of transmission, it is not possible to release any of the $r_f^{j-1} + \text{N}_2$ buffers reserved for flow f at time t_f^j. Hence, for all flows $f \in H$, the buffer allocation and reservation is given by:

$$\text{A}_f^j = \begin{cases} (r_f^{j-1} + y_f + \text{N}_2) & \text{if } y_f \le 0 \\ \left(r_f^{j-1} + \left\lfloor \frac{\max\{0, y_f\} * \mathcal{A}}{\sum_{f \in H} \max\{0, y_f\}} \right\rfloor + \text{N}_2\right) & \text{if } y_f > 0 \end{cases} \tag{13}$$

and

$$r_f^j = r_f^{j-1} + \left\lfloor \frac{\max\{0, y_f\} * \mathcal{A}}{\sum_{f \in H} \max\{0, y_f\}} \right\rfloor \tag{14}$$

Whereas Equation (13) reduces the credit allocation of all the flows $f \in H$ for which $y_f < 0$ (and thereby takes a step towards achieving fair distribution of buffers), it enhances the credit allocation of all the remaining flows in H as per their respective increases in buffer allocations. Equation (14), on the other hand, ensure that: (1) the buffer reservations for all flows $f \in H$ for which $y_f \le 0$ are not altered, and (2) the available buffers (namely, $\mathcal{A}$) are fairly distributed among all the flows $f \in H$ for which $y_f > 0$.

2.2.3 Playback Adaptation Algorithm

The buffer allocation algorithm and bandwidth estimation techniques presented in the previous section enable the receiver-based credit flow protocol to quickly adapt to the changes in the bandwidth requirements of video sources, and thereby minimize the end-to-end delay. However, due to its inherent nature, the protocol does not provide any bounds on the delay. Consequently, once the playback of a video stream is initiated at the destination site, a frame arriving later than its scheduled playback instant will result in playback discontinuity. To provide a good quality of service, the destination site must employ playback point adaptation algorithm, which minimizes: (1) the number of playback discontinuities resulting from variations in end-to-end delay, and (2) the effective end-to-end delay (defined as the difference between time at which a frame is captured at the source and the time at which it is displayed at the destination). Whereas the former objective can be attained by buffering, and hence delaying, the frames for sufficiently long durations at the destination prior to their playback (i.e., by introducing anti-jitter delay), the latter can be met by scheduling the the playback at the earliest.

To formulate a policy for balancing these two antagonistic requirements, let us denote the arrival time of first frame at the destination and the anti-jitter delay by a_1 and D, respectively. Hence, the playback will be initiated at the destination at time $p_1 = a_1 + D$. Moreover, if I denotes the inter-frame separation, then the time instant at which the j^{th} frame ($j \ge 2$) must be displayed is given by $p_j = p_1 + (j-1) * I = a_1 + (j-1) * I + D$.

Depending on the relationship between the arrival time of a frame and its scheduled playback instant, the playback adaptation algorithm must handle the following two cases:

- $a_j > p_j$: In this case, a playback discontinuity will be observed at the destination. In such a scenario, to prevent frequent occurrences of such playback discontinuities, the playback adaptation algorithm may increase the anti-jitter delay. That is, rather than displaying the j^{th} frame as soon as it arrives, the algorithm may display the frame at time $p'_j = a_j + \theta_1$, ($\theta_1 > 0$), thereby increasing the anti-jitter delay to

$D' = p'_j - (a_1 + (j-1) * I)$.

- $a_j < p_j$: This condition is indicative of reduced congestion in the network and the adaptation algorithm exploits it to reduce the effective end-to-end delay by discarding a frame. It can use a first order auto regressive filter to estimate the reduction in the network delay and discard a frame when the estimator indicates the reduction to be greater than θ_2 ($\theta_2 > I$). The algorithm can exploit compression specific information in order to construct a good estimator and decide which frame to discard. For example in MPEG encoded video flows, the maximum end-to-end delay is experienced by I frames and hence the estimator may use the arrival and playback instants of these frames only. Moreover, it can discard only a B frame to avoid affecting the decoding of any other frames.

2.3 Discussion

The algorithms that we have presented multiplex the short duration bursts such that packet losses do not occur while minimizing buffer requirement and end-to-end delay. However, when sustained long term bursts occur and the aggregate bandwidth requirement of the sources exceeds the link capacity, the queues at the sources will build up and end-to-end delay would increase. In such a scenario, an application can reduce the effective end-to-end delay by invoking application specific procedures to reduce the spatial, temporal or chroma resolution. However, to limit the occurrence of such an event, a network must employ admission control algorithm. Since the protocol guarantees that no packet loss occurs even during congestion, heuristic admission control algorithms based on measured traffic statistics will suffice [2]. Thus, our protocol not only effectively controls the short term burstiness, but also simplifies the network control for long term burstiness.

3 Experimental Evaluation

We have experimentally evaluated various parameters and aspects of our video transmission protocol through extensive trace-driven simulations. These simulations were carried out using an enhanced version of the REAL network simulator available from the University of California at Berkeley. To evaluate the various parameters of the protocol, we have experimented with several network topologies. For the most part of this section, we will present our simulation results for a *baseline* network topology consisting of 6 switches (see Figure 1) which stresses various aspects of the protocol and is similar to the topology used in [5]; results obtained from other topologies are summarized in

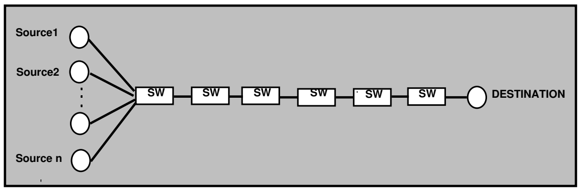

Figure 1 : The baseline network topology for the simulations

[3]. In our topologies, all the links are assumed to be duplex, of capacity 40 Mb/s, and with a propagation delay of 3 ms. Finally, to simulate an ATM network environment, the packet size was chosen to be 53 bytes (5 bytes of header information and 48 bytes of payload).

Each flow was simulated to carry a VBR encoded video stream, the bit rate traces for which were obtained from 3 MPEG and 1 JPEG encoded video sequence. Each video source transmits a randomly selected part of one of these video sequences to the destination node. We used two *baseline* configurations of sources which induce different load on the switches. In *configuration 1*, there were total of 8 sources transmitting their I frames out of phase, i.e., the sources were not synchronized. In *configuration 2*, there were a total of 15 sources transmitting their I frames simultaneously (the sum of maximum of average bit rate requirements derived over 1 second intervals was 40 Mb/s). For most of our experiments, the buffer size at the switch and the adaptation interval were set to 1000 packets and 30 ms, respectively. The network was simulated for the duration of 10 seconds.

3.1 Bandwidth Estimation

To validate our hypothesis that the rampup factor is a function of the burstiness of the video source, we examined the effect of varying rampup factors on the end-to-end delay observed by the JPEG and MPEG sources as well as the buffer space requirement at the switches. As argued in Section 2.2.1, increase in the rampup factor decreases the end-to-end delay, but imposes larger buffer space requirement (see Figure 2). Figure 2 also depicts the relative performance of the techniques for choosing fixed and frame-specific rampup factor for MPEG flows. Since an analysis of our MPEG traces indicated that B frames are the smallest in size, P frames are approximately twice as large as B frames, and I frames are approximately 6-10 times as large as B frames; we chose $\gamma_B = 1$ and $\gamma_P = 2$, and then studied the effect of increasing the value of the rampup factor for I frames (namely, γ_I) on the end-to-end delay and buffer space requirement. As Figure 2 illustrates, selecting $\gamma_I = 6$ yields the smallest maximum end-to-end

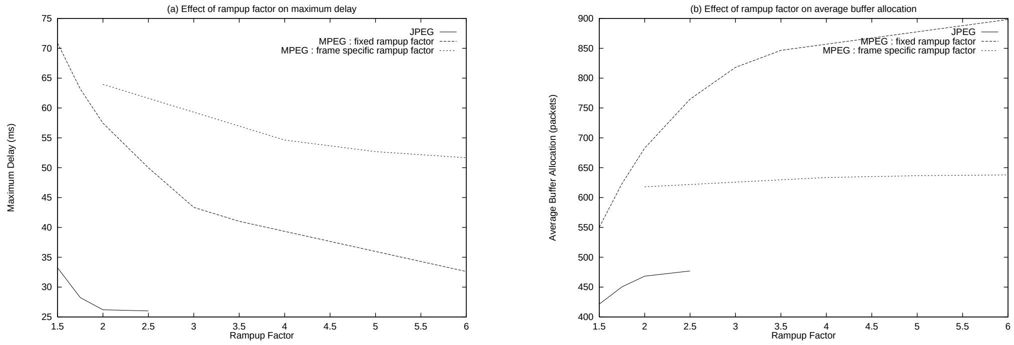

Figure 2 : Effect of rampup factor on: (a) maximum delay and (b) average buffer space requirement

delays. Moreover, the maximum delays observed with $\gamma_I = 6$ were approximately the same as those observed for the fixed rampup factor case with $\gamma = 2.5$. The main advantage of employing frame-specific rampup factor is that it provides better delay-jitter performance and requires smaller amount of buffer space ($\approx 25\%$ lower as compared to the fixed rampup factor case with $\gamma = 2.5$ - see Figure 2(b)). Hence, for the rest of this section, we assume that the frame-specific rampup factor scheme is employed for MPEG flows.

As described in Equation (2), the bandwidth requirement of a flow during an adaptation interval is estimated using a priori known minimum and maximum bandwidth requirements (namely, BW_f^{min} and BW_f^{max}). Our experiments have demonstrated that if BW_f^{max} is chosen conservatively, then neither of the parameters effect the end-to-end delay.

3.2 Effect of Configuration Parameters

The smaller the adaptation interval, the quicker is the process of adapting to changes in bandwidth requirements, and hence, the smaller is the delay. However, small values of adaptation interval induces large network overhead. To help evaluate the tradeoff in selecting the adaptation interval, we studied the variation in the maximum end-to-end delay with increase in the adaptation interval (see Figure 3(a)). As expected, it illustrates that the end-to-end delay increases linearly with increase in adaptation interval. Since the transmission of a frame can begin at any instant within an adaptation interval, the expected duration for adapting to the change in the bandwidth requirement of a frame is half of adaptation interval. Hence, as depicted in Figure 3(a), with each increase of ϵ in the allocation interval, the maximum delay increases by about $\frac{\epsilon}{2}$. For video streams being played back at 30 frames/sec, an adaptation interval of $\approx$ 30 ms is sufficient.

An important switch parameter that effects the delay performance of sources is the buffer size at each of the switches. Figure 3(b) demonstrates that the delay performance improves as buffer size increases. Moreover, it illustrates that the buffer size of about 900 packets is sufficient to provide a good delay performance for the video sources. Note that the experimentally observed buffer space requirement is about 20% larger than that required for sustaining a bandwidth of 40Mb/s on a link with 3 ms propagation delay in this configuration. In comparison, if the *exact* peak bandwidth requirements of the sources were known and if a non-adaptive credit allocation algorithm was used, a buffer of size 1642 packets would have been required for this configuration. Hence, non-adaptive credit management would have required 64.2% more buffer.

In adaptive buffer management, in order for the buffer reallocation at the first switch to be effective, the available buffers at each of the downstream switches must be appropriately reallocated. Consequently, the number of switches along the path from the source to the destination of a flow may impact the maximum delay yielded by the adaptive buffer management algorithm. To precisely quantify this dependence, we simulated network topologies with number of switch varying from 2 through 6, and analyzed each topology with the traffic induced by configuration 2. As Figure 4(a) demonstrates, although the maximum delay increases with increase in the number of switches, the increase is not significant. In fact, the increase in the end-to-end delay observed when the number of switches increased from 2 to 6 is approximately equal to the propagation delay between switch 2 to 6 (i.e., $\approx$ 12 ms).

3.3 Effect of Load

One of the main goals of the adaptive buffer management algorithm is to ensure graceful degradation in the delay performance with increase in the network load. To evaluate the effectiveness of our algorithm with respect to this criteria, we measured the maximum delays observed by the sources at varying utilization when (1) their I frames were

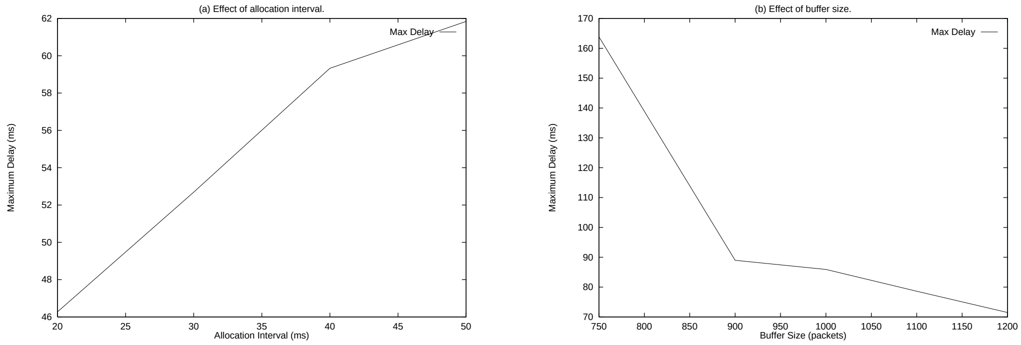

Figure 3 : Effect of increasing: (a) allocation interval and (b) buffer size on maximum delay

aligned, and (2) they were out of phase. Since the maximum delays are determined by the peak utilization of a link, Figure 4(b) plots the variation of maximum delay with the peak utilization. The peak utilization was chosen to be the maximum average utilization, with the averaging interval being 1 second. As the figure illustrates, sources experience lower delays at lower loads. Moreover, the sources experience lower delay when the sources are out of phase than in phase, thereby demonstrating that the buffer allocation is able to effectively allocate bandwidth at varying loads.

3.4 Playback Adaptation Algorithm

To evaluate the effectiveness of the playback point adaptation algorithm presented in Section 2.2.3, we simulated our network topology with configuration 2 load for *60 seconds*. In our experiment, only 4 playback discontinuities across all the sources were observed during the entire simulation, and that the algorithm was able to adapt to decrease in end-to-end delay by discarding frames. Moreover, the average length of the discontinuity was less than 2.5 ms which is within human perceptual tolerance.

4 Comparison with Other Schemes

4.1 Receiver-Oriented Adaptation

The concept of receiver-oriented adaptation was originally introduced in [6], and a particular algorithm was proposed in [7]. The algorithm presented in [7] requires $\sum_{f \in S} \mathrm{A}_f^j \leq \frac{M}{2} - \sum_{f \in S} Q_f(t_f^j)$ to hold after the j^{th} allocation. It is easily observed that the condition for reliability presented in this paper is much weaker and consequently leads to much more efficient utilization of network resources. Our experiments have demonstrated that for configuration 2 and a buffer size of 1000 packets, the link utilization in our algorithm is 60-70% higher. Moreover, the end-to-end delay yielded by our algorithm is 20 times smaller (85 ms as compared to 1646 ms). Furthermore, the algorithm in [7] requires 100% more buffers (2000 packets) for a comparable delay performance. Since peak rate buffer allocation of the sources would have required 1650 packets, we believe this is an anomaly in the algorithm in [7]. Finally, when no flow control algorithm is employed the buffer requirement is 2320 packets. Hence, the algorithm in [7] increases the delay significantly while saving very small amount of buffer space for video flows. This also illustrates that the adaptive credit-based protocol would not be feasible for video without Theorem 1.

The algorithm presented in [7] presumes additional per link timers at each of the upstream nodes. Moreover, these timers are assumed to be synchronized. We believe that to facilitate interoperability, adaptive buffer management should be an implementation choice of a switch and should not require any additional support from other switches. This assumption is unnecessary and, as we have demonstrated, reliability can be achieved without additional support from the upstream switches. Furthermore, the algorithm in [7] permits buffer adaptation for the flows only at every adaptation interval, which makes the algorithm inflexible. This is in contrast to Theorem 1, which does not assume any correlation between the time instants at which adaptation occurs for the flows.

4.2 Video Transmission Schemes

We chose the following three schemes, which we believe are representative of the schemes proposed in the literature, for the purpose of comparison:

- *Periodic Averaging*: This is a source traffic shaping mechanism [13] in which, assuming all the frame sizes are known in advance, all the frames that belong to an averaging interval are transmitted at the average rate for that interval. This generates a smooth traffic source with rate changes occurring only at averaging interval boundaries. The delay incurred by frames

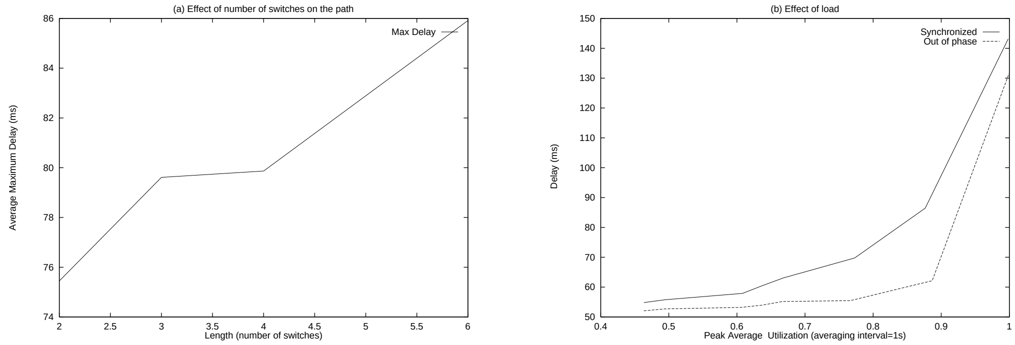

Figure 4 : Effect of increasing: (a) number of switches on the path and (b) load on maximum delay

in this scheme would be a lower bound on the delay that would be introduced at the source to generate an equivalent smooth source when the frame sizes are not known. For the purpose of comparison, we chose an averaging interval of 1 second.

- *Frame Smoothing*: In this scheme, after a frame is completely encoded, it is transmitted at a rate such that the transmission is completed within an interframe duration, i.e., before the next frame is encoded. This scheme reduces burstiness caused due to the encoding process but introduces smoothing delay equivalent to encoding delay. We assume encoding delay to be interframe delay.

- *Rate Based Control*: In this scheme, rate of transmission of a video source is controlled by the rate based congestion control protocol proposed in ATM forum. Since the ATM forum has not specified a particular switch behavior, we assume the switches to be EFCI switches which mark *resource management* packets on both the forward and reverse paths.

We selected maximum delay, maximum queue length and packet loss (when the buffer size is limited to 1000 packets) as the metrics for comparison with other schemes. Figures 5 and 6 plot these metrics at varying maximum average utilization level (averaging interval is chosen to be 1 second). Figure 5(a) shows that delay in our protocol is approximately 50% of the delay in periodic averaging and almost the same delay as in frame smoothing. Figure 5(b) demonstrates that the buffer requirement in our protocol is significantly smaller than that required when frame smoothing is employed. Hence, we conclude our protocol finds a middle ground between the two extremes of source traffic shaping and no traffic shaping.

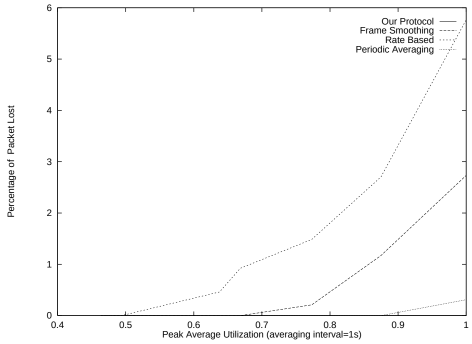

Figure 6 : Comparison of packet loss with other video transmission schemes

5 Concluding Remarks

We presented a network layer protocol which effectively multiplexes VBR encoded video traffic. The protocol minimizes buffer requirement as well as end-to-end delay and guarantees that packet losses do not occur. To minimize buffer requirement as well as avoid packet losses, we derived necessary and sufficient buffers required for ensuring the reliability of the receiver-based adaptive buffer allocation algorithm. We then presented bandwidth estimation techniques and buffer allocation algorithm which are specifically designed for video and minimize end-to-end delay and jitter for VBR encoded video. We defined a new fairness criteria for buffer allocation which ensures uniform increase in the delays experienced by the video frames during congestion. Finally, we presented a playback point adaptation technique.

We experimentally evaluated the performance of the algorithm under a wide range of parameters and demonstrated its suitability. We demonstrated that the previously known receiver-oriented buffer adaptation algorithm not only requires 100% more buffers and incurs 20 times higher delay as compared to our algorithm, but is also not viable

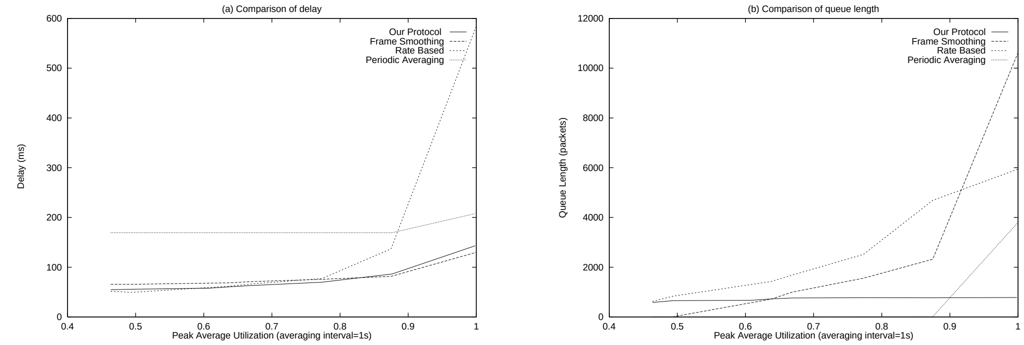

Figure 5 : Comparison of delay and queue length with other video transmission schemes

for video transport. We demonstrated that the protocol does not require a priori specification of the source traffic characteristics. We also demonstrated that since network, rather than source, shapes the traffic, the protocol provides significantly better delay performance than source traffic shaping mechanisms while requiring significantly smaller buffer than protocols which do not employ traffic shaping. Though the protocol does not guarantee delay, the clients are able to realize smaller delay than guaranteed delay and a reliable service by using the playback point adaptation technique. The protocol adapts to the network load and provides service that commensurates with the network congestion. In summary, we have demonstrated that our protocol is a viable alternative to guaranteed rate transmission of video for applications which do not require the network to provide strict performance guarantees.

REFERENCES

[1] E.W. Biersack. Performance Evaluation of Forward Error Correction in ATM Networks. In *Proceedings of ACM SIGCOMM'92, Computer Communications Review, Vol. 22, No. 4*, pages 248–257, August 1992.

[2] D.D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network. In *Proceedings of ACM SIGCOMM*, pages 14–26, August 1992.

[3] P. Goyal, H. M. Vin, C. Shen, and P.J. Shenoy. A Reliable, Adaptive Network Protocol for Video Transport. Technical Report TR-95-18, The University of Texas at Austin, July 1995. Available via URL http://www.cs.utexas.edu/users/dmcl.

[4] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. In *Proceedings of ACM SIGCOMM'95*, 1995.

[5] H. Kanakia, P. P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video. *Proceedings of ACM SIGCOMM Computer Communications Review*, October 1993.

[6] H. T. Kung, T. Blackwell, and A. Chapman. Credit-based Flow Control for ATM Networks : Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing. In *Proceedings of SIGCOMM'94, London*, pages 101–114, August 1994.

[7] H. T. Kung and K. Chang. Reveiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks. In *Proceedings of INFOCOM'95*, April 1995.

[8] H.T. Kung. The FCVC(Flow-Controlled Virtual Channels) Proposal for ATM Networks: A Summary. In *Proceedings of the INFOCOM'93*, 1993.

[9] S.S. Lam, S. Chow, and D.K.Y. Yau. An Algorithm for Lossless Smoothing of MPEG Video. In *Proceedings of ACM SIGCOMM'94, London*, 1994.

[10] C. Ozveren, R. Simcoe, and G. Varghese. Reliable and Efficient Hop-by-Hop Flow Control. In *Proceedings of SIGCOMM'94, London*, pages 89–100, August 1994.

[11] P. Pancha and M. E. Zarki. MPEG Coding for Variable Bit Rate Video Transmission. *IEEE Communications Magazine*, pages 54–66, May 1994.

[12] C.J. Turner and L.L. Peterson. Image Transfer: An End to End Design. In *Proceedings of ACM SIGCOMM'92, Baltimore*, pages 258–268, August 1992.

[13] G.A Veciana. *Design Issues in ATM Networks: Traffic Shaping and Congestion Control*. PhD thesis, University of California at Berkeley, 1993.