This is the author's version of an article that has been published in IEEE Transactions on Multimedia. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2002.806537
M. Mielke, R. S. Aygun, Yuqing Song and Aidong Zhang, "PLUS: a probe-loss utilization streaming mechanism for distributed multimedia presentation systems," in IEEE Transactions on Multimedia, vol. 4, no. 4, pp. 561-577, Dec 2002.

Second revision

10/6/2001

PLUS: A Probe-Loss Utilization Streaming Mechanism for Distributed Multimedia Presentation Systems *

Markus Mielke,† Ramazan Savaş Aygün, Yuqing Song and Aidong Zhang
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260

Abstract

In this article, we present a new flow and congestion control scheme, PLUS (Probe-Loss Utilization Streaming protocol), for distributed multimedia presentation systems. This scheme utilizes probing of the network situation and an effective adjustment mechanism to data loss to support multimedia presentations. The proposed scheme is also designed to scale with increasing number of PLUS-based streaming traffic and to live in harmony with TCP-based traffic. The novelty of the PLUS protocol is that it utilizes the knowledge of its future bottleneck bandwidth in probing the current network situation. This can be achieved by *a priori* knowledge of the multimedia data before a presentation is requested by a client. Compression schemes like MPEG introduce dependencies on media units. I frames are needed to successfully decode P and B frames, and P frames are needed to decode B frames. A loss of an I or P frame automatically eliminates dependent media units. Our probing scheme increases the successful transmission of critical I and P packets without the overhead of error-correction-schemes. Probing is done using B-frame packets. The advantage is that we use data packets as probe packets. With the PLUS protocol we address the need to *avoid* congestion rather than *react* to it. Experiments demonstrate the effectiveness of the approach in utilizing network resources and decreasing loss ratios.

Keywords: multimedia streaming, congestion control, probing, multimedia presentation systems

1 Introduction

There has been increasing interest in flexibly constructing and manipulating heterogeneous presentations from multimedia data resources [29, 6] to support sophisticated applications [42, 23, 1, 17]. In these applications, information from multimedia data resources at one location must be made available at other remote locations for purposes of collaborative engineering, educational learning and tutoring, interactive computer-based training, electronic technical manuals, and distributed publishing. Such applications require that basic multimedia objects be stored in multimedia databases/files. These multimedia objects, such as audio

*This research is supported by an NSF CAREER grant IIS-9733730, NSF Digital Government and Digital Libraries programs.
†Current address: Microsoft Corporation, One Microsoft Way Redmond, WA 98005

samples, video clips, images, and animation, are then selectively retrieved, transmitted and composed for customized presentations. Multimedia objects, such as audio samples, video clips and transparencies, are characterized by large volumes and temporal constraints. Compression techniques [18] help reduce these volumes but they also introduce burstiness into the streams [10]. Bursty streams with varying bandwidth requirements increase the unpredictability of congestion in current networks.

The current network architecture, as embodied in the IP [8] network protocol, was designed for data-oriented applications and does not provide useful packet delivery service for interactive multimedia streams. A large part of multimedia applications currently in the Internet, such as VIC [28],VAT [24] and Netshow, are based on the new RTP (real-time-transport) protocol [34]. RTP does not provide services that are normally provided by a transport protocol. That is, it offers no reliability mechanisms, has no understanding of a connection and is usually implemented as part of the application relying on UDP (part of the IP stack) as the transport protocol. Instead RTP offers the application the capability of distinguishing between different media streams and keeping track of various statistics describing the quality of the session. However, both UDP and RTP offer no quality of service control mechanism. Fluctuations of the network conditions combined with the inability of those protocols to support quality of service (QoS) control often renders multimedia applications useless.

UDP and RTP as non-adaptive streaming protocols also have a noticeable effect on other applications in the network, especially when competing with adaptive protocols such as TCP. Under heavy load, TCP will back off, reducing its bandwidth utilization, while non-adaptive streams will continue to push their loads through bottlenecks without considering their neighbors. The effect of this behavior is to drive up the packet drop rate and will lead to starvation in neighboring TCP and packet-loss sensitive streams [27].

To combat rising packet loss rates, the Internet Engineering Task Force (IETF) is considering widespread deployment of active queue management techniques to improve the performance of congestion responsive applications and to punish non-adaptive ones. The techniques being considered are based on Random Early Detection (RED) [15]. RED punishes streams by dropping packets randomly out of their waiting queues. IETF employs RED along with additional mechanisms to identify malicious flows and force them to maintain a fair share of the available bandwidth. These mechanisms have a catastrophic impact on motion predicted compression schemes like MPEG [12], which are used for video data compression.

Compression schemes like MPEG introduce dependencies among media units. I frames are needed to successfully decode P and B frames, and P frames are needed to decode B frames. Only B frames have no dependencies. A loss of an I or P frame automatically eliminates dependent media units. Therefore, we define packets belonging to I and P frames as *critical packets*. The aim of a multimedia streaming protocol must include protection of critical packets.

The loss of critical packets has a high impact on the quality of service, it might even lead to complete

distortion of a media stream [39]. Transmission support for multimedia presentation systems, which do not seriously consider adaptation to loss rates can contribute to a widespread congestive collapse in the Internet [12] and risk punishment with the establishment of the RED router scheme.

In this article, we present a new flow and congestion control protocol scheme, PLUS ((Probe- Loss Utilization Streaming protocol), for distributed multimedia presentation systems. This scheme utilizes probing of the network status and an effective adjustment mechanism to data loss to support multimedia presentations. With the PLUS protocol we address the need to *avoid* congestion rather than *react* to it. By probing we mean a mechanism to test the network with current data packets to see if the maximum bandwidth requirement (introduced by the compression scheme) in the future can be supported. Our scheme gives the application time to prepare countermeasures like backing off streaming rates or conducting frame dropping for video streams if the probing gives information that the future data rate can not be supported.

The presented scheme is also designed to scale with increasing number of PLUS-based streaming traffic and to live in harmony with TCP-based traffic. PLUS offers the following features:

- Probing: the network will be probed to test if the maximum bandwidth requirement of a stream can be supported.
- Better protection of I and P frames.
- Just-in-time delivery: media units are only sent out depending on their timing requirements, minimizing pre-fetch buffer requirements at the client.
- Smoothed transmission: packets within a given window are spread out to reduce bursty bandwidth requirements introduced by compression techniques (such as MPEG).
- TCP friendly adjustment scheme: decrease of bandwidth utilization in case of congestion. If needed, media dropping at the server site is initiated to avoid random packet dropping by routers, and to avoid starvation of neighboring TCP connections.

The rest of this article is organized as follows. Section 2 discusses the related work. In Section 3 the PLUS architecture is formulated. Section 4 presents the implementation of the *NetMedia* transport scheme and the experiment evaluating the PLUS protocol. Concluding remarks are offered in Section 5.

2 Related Work

Research has been conducted in three different areas to handle congestion control in the network. The first is to protect audio/video streams from the effects of congestion by reserving resources (e.g., buffers and CPU cycles at a router, bandwidth at the network). The aim is to guarantee predictable levels of services.

The RSVP architecture [45] provides receiver-initiated reservations to accommodate heterogeneity among receivers as well as dynamic membership changes. Such mechanisms, like RSVP, have significant scalability problems.

The second area, the so called best effort approach, assumes no direct support for resource reservation in the network and attempts to adapt the media streams to current network conditions. Best effort transmission schemes adaptively scale (reduce or increase) the bandwidth requirements of audio/video streams to approximate a connection that is currently sustainable in the network.

One approach is to measure and monitor the available bandwidth at any given time. The idea is to probe the network by sending probing packets at the flow rate of the data stream. Decisions are made based on the packet loss of the probing stream. Flows will be adjusted based on a certain threshold. Examples of this technique can be found in [25] and [41]. The disadvantage of this approach is that probing packets are kept small and do not necessarily reflect the required bandwidth of a stream. Also, decisions about flow rates are based on the past performance of the network and do not reflect the current behavior at the time of streaming the data.

Another best effort approach is to modify the TCP protocol. In [9], the authors propose to use TCP without its retransmission scheme for streaming applications. Removing the retransmission part from TCP eliminates the resultant problems of latency and wasted network bandwidth. But due to TCP's congestion control algorithm, the streaming remains inherently bursty. The SCP (streaming control protocol) presented in [7] reduces the burstiness of TCP without retransmission. SCP uses a *direct streaming* approach, which utilizes the maximum bandwidth available in the network for streaming. This increases the risk of congestion caused by SCP streams. The advantage is, in case of congestion, SCP backs-off similarly to TCP. Since there is no need to send out multimedia data faster than its timing requirement, the drawback of direct streaming can be avoided.

The goal of a model-based approach is to establish an analytical model of TCP traffic. This will allow to calculate the available bandwidth given certain boundary conditions. Floyd and Ott [16, 40] propose a rate-based flow control scheme based on the analysis of TCP throughput. The proposed model estimates the throughput of a TCP connection under known delay (round trip time) and loss conditions. Based on this estimation the system restricts its transmission rate to the throughput of an equally competing TCP connection. The advantage of this model is its simplicity. This also leads to its disadvantage because the model does not consider timeout cases or delayed acknowledgments. It is also based on *average* loss and delay observations. However, adaptation decisions need to be taken based on the current loss and delay values. Using the TCP throughput model as the sole basis for adaptation results in a rather oscillatory adaptation behavior. Different studies [35] have also shown that this model is only accurate enough for losses of less than 16%. The sending rate of model-based protocols may drop to 0 at high loss rate which is undesirable for multimedia applications [19].

Sending best-effort traffic without any consideration of the network congestion state caused by TCP and non-TCP traffic can easily increase the risk of packet losses. Therefore, the aim is to develop adaptive schemes, which not only target at loss ratios and bandwidth utilization but are also fair towards competing TCP connections. A good example is the direct adjustment algorithm (DAA) [35]. It is based on a combination of two approaches described in the literature, namely: additive increase/multiplicative decrease schemes proposed in [3, 5] and an enhancement of the TCP-throughput model described in [16].

TCP uses additive increase, multiplicative decrease mechanism (AIMD) to detect additional bandwidth and to react to congestion. RAP [31] uses a simple AIMD mechanism where each packet is acknowledged by the receiver. The increase rate is one packet per round-trip time and when congestion is experienced the sending rate reduces to its half. Simple AIMD is the most efficient and best suited algorithm for bulk data transfer operations that can tolerate large reductions in available capacity upon encountering congestion [2]. Simple rate-based AIMD schemes such as RAP have the same large variations in data rate as TCP [20]. So, it is not suitable to use it (as in TCP) for multimedia applications. LDA+ [36] adjusts increase and decrease factors for AIMD dynamically to network conditions.

TCP emulation at receivers (TEAR) protocol [19] computes the sending rate at the receivers. The receivers compute the feedback rate based on weighted average of a number of epochs. The sender sets its current transmission rate to the most recent estimated rate sent by the receiver. Since the rate is computed at the receiver, TEAR refrains from acknowledging each packet. This reduces the congestion in reverse path for asymmetric networks such as wireless networks, cable modems. A survey on TCP-friendly congestion protocols compares most of the protocols [20]. The performance of some of these protocols have been tested under various conditions to check fairness, aggressiveness, responsiveness and smoothness [43]. Each protocol has its own drawback and does not dominate other protocols in all aspects.

The third approach towards congestion control referred to as *active queue management* is to establish policies in the routers to punish not well behaved streams. One form of active queue management has been proposed by the IETF called RED (Random Early Detection) [4, 13]. RED maintains an exponentially-weighted moving average of the queue length, which it uses to detect congestion. When the average queue length exceeds a minimum threshold, packets are randomly dropped or marked with an explicit congestion notification bit [13, 32, 33]. When the average queue length exceeds a maximum threshold, all packets are dropped or marked. An implicit assumption behind the design of RED is that all flows respond to loss as an indicator of congestion. The idea is that unresponsive streams will be punished harder by experiencing a higher loss rate than responsive streams. The disadvantage is that unresponsive streams may dominate a router's queue. Lin and Morris recognize this shortcoming of RED and proposed a scheme called Flow Random Early detection (FRED), to promote fair buffer allocation between flows [26]. In [30], Jeffay and Smith introduced CBT (Class-Based Thresholds). The idea is to provide the congestion avoidance benefits of RED while providing protection for TCP and well-behaved UDP flows.

Current transmission schemes need to take into account the congestion avoidance mechanisms introduced by the router schemes in order to be recognized as *well behaved* streams. Punishment by some routers has a drastic effect on the QoS of multimedia streams, especially if critical packets from I or P frames are randomly dropped during the punishment process. Protecting critical packets is therefore a key factor in increasing the QoS of a presentation. The PLUS algorithm presented in this article addresses these needs by enhancing the survival rate of critical packets without increasing the overhead introduced by traditional forward-error-correction schemes.

3 PLUS Architecture

A unicast streaming scenario with the PLUS protocol consists of media servers and a media client, linked by a network. Media packets with a constant packet size are streamed with respect to their time-dependency from the server (sender) to the client (receiver). Each packet carries among other fields its sending time and

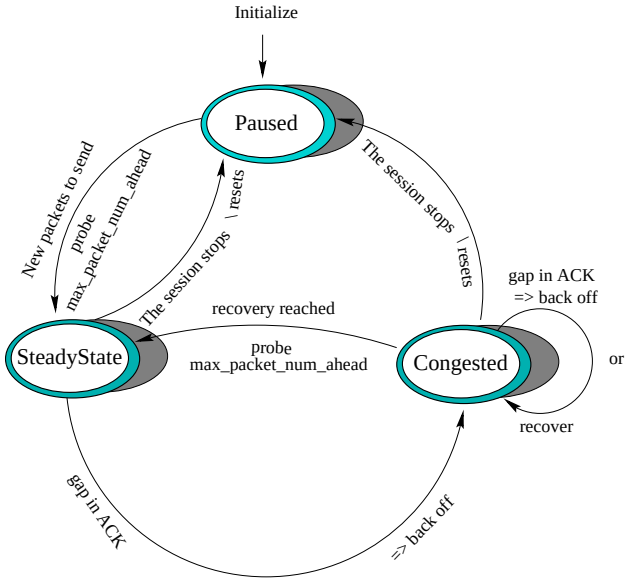

Figure 1: PLUS state transition diagram.

an incremental sequence number. In case the PLUS probing scheme indicates congestion, the receiver issues a feedback message to the server. Feedback messages are generated using a TCP connection to guarantee arrival at the server. Based on the packet loss within an observation window ($\triangle W$), the client sends an adjustment request to influence the sending rate of the stream to behave TCP friendly and to avoid further network congestion.

The PLUS protocol consists of three states (see Figure [1]): *Paused*, *SteadyState* and *Congested*. Each state is associated with a specific condition of the network and a transition policy. After initialization of the server the system enters the paused state. After a request for new packets arrives at the server the PLUS

protocol starts streaming with the probing algorithm activated. The protocol is now in the SteadyState mode, probing the available bandwidth of the network while streaming the data to the client. When a packet loss is detected, PLUS enters the congestion state. After detection of packet loss at the receiver, the best adjustment rate for sending rate with the congestion information is sent as an acknowledgment to the server. Packets are transmitted in order and timeouts are based on the round trip time.

In case of a packet loss in the observation window, the protocol performs a multiplicative back-off policy that reduces the bandwidth to live in harmony with neighboring TCP traffic. If the back-off policy violates the schedule of the time-dependent media units, a prioritized media dropping at the server site is initiated. This will reduce the quality of the presentation as minimally as possible but will allow the presentation to continue even in the case of congestion. After reaching a bandwidth the network can support, the PLUS protocol tries to recover by additively increasing the sender rate until the steady state is reached again.

3.1 Smoothing and Probing Algorithm

Data compression can introduce burstiness into data streams. The burstiness results in different bandwidth requirements during transmission of the stream. This makes it difficult to come up with a resource and adaptation scheme because bandwidth requirements always change.

The PLUS protocol eases bandwidth fluctuations by grouping together some number of media units in a window (time) interval $\triangle W$ (e.g. assigning 150 frames of video to 5-second $\triangle W$). Reducing the burstiness of a stream in a given window by spreading its packets equally is defined as *smoothing* [11]. Smoothing is done by sending out the data at the *average bandwidth* requirement for the window. The average bandwidth is equal to spreading the packets uniformly over the $\triangle W$ interval and sending at a fixed rate $\triangle a$ (Equation 1).

$$\triangle a = \frac{\triangle W}{number\_of\_packets\_in\_current\_interval}. \tag{1}$$

By using this method, clients can guarantee that the bandwidth needed is minimal and constant through interval $\triangle W$. The disadvantage of this method is that a bigger prefetch buffer is required to assemble frames of a large size.

At each presentation interval, we identify a *critical interval.* The critical interval is an interval in the future playback time that contains the maximum number of packets. The aim of the PLUS probing scheme is to test if the network can support the critical interval. The advantage is that the PLUS protocol does not send out those packets, which may be lost or cause congestion, and then tries to adapt to this situation. It tests ahead of time if sending data at the rate required by the critical interval is feasible or not.

The size of $\triangle W$ is a trade off between the amount of smoothing and the delay caused by prefetching. A large interval $\triangle W$ will require a large buffer and large set-up times before a stream can be displayed because for dependent media units (like B-frames), all necessary packets (from I or P frames) need to arrive first at

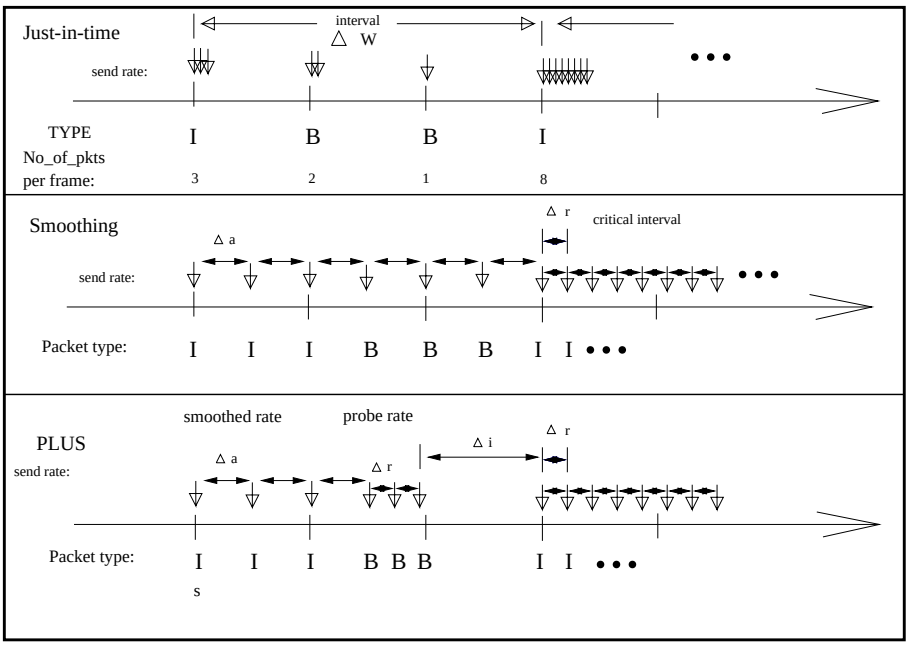

Figure 2: PLUS probing scheme.

the client site before they can be decoded.

One choice is to set $\triangle W$ to the length of time for sending one frame. The advantage is that the probing scheme collects information if critical packets (from I-frames), which will be in the intervals with the highest number of packets, can be supported. In the presence of congestion, the PLUS protocol will adapt to increase the probability of a successful transmission of critical packets. The disadvantage is that the PLUS protocol can apply smoothing only to the packets within a frame. For smoothing algorithms, it is more feasible to group a larger number of frames together to lower the peak bandwidth required.

Another choice is to set $\triangle W$ to a short sequence of frames such as five to ten seconds of playback. The advantage is that buffer management still can easily handle such a data size (10 seconds equals 300 frames at 30 frames per second). The probing scheme will then identify stream sequences with higher rates of change, like video scenes with a lot of movement, and proactively test the feasibility of sending at these higher rates.

As a reasonable trade off for $\triangle W$, we suggest for video streams the use of five seconds smoothing. We base our decision on the knowledge that the human awareness of frame losses in scenes with movement is much higher than in static scenes. If the PLUS probing scheme detects congestion, countermeasures like increasing the smoothing window or in the worst case packet dropping can be initiated to adapt to the congestion in the network and protect critical intervals.

In the first graph of Figure 2, we see the normal just-in-time sending schedule of a multimedia presentation stream. In each schedule interval within $\triangle W$, the system sends out a number of constant sized packets to fulfill the timing requirements for each slot. The data in each interval is normally not smoothed and will

be sent out as fast as possible. This will cause short bursts of high data transmission rates, especially for I and P frames.

In the average smoothing scheme (the second graph in Figure 2), packets within $\triangle W$ are evenly spaced apart by $\triangle a$. Each interval $\triangle W$ has its own $\triangle a$, which is dependent on the number of packets that comprise $\triangle W$. The goal is to find the future interval that has the smallest $\triangle a$. We define the corresponding $\triangle W$ as the *critical interval* of the current $\triangle W$. The critical interval defines the bottleneck of the stream.

Let's assume the critical interval for s is the next interval, which starts with an I frame of 8 packets. The advantage of multimedia presentation systems is that media streams are known a priori to their transmission. This feature can be used to save computational time in finding the maximum required bandwidth (critical interval) for each given interval by *preprocessing* the media streams.

interval number	n+1	n+2	n+3	n+4	n+5	n+6	
pkts in current $\triangle W$	2	3	8	4	2	5	EOF
pkts in future critical interval	8	8	8	5	5	5	

Table 1: Preprocessing data.

The preprocessing (Table 1) is done by starting at the end of the stream (EOF) and moving back to the beginning, checking each interval on the way. For each interval we keep an array entry, which records the number of packets for its critical interval. After the preprocessing is done, the array contains for each interval $\triangle W$ its corresponding critical interval which will be the bottleneck of the stream some time in the future.

Once we determine the critical interval, we apply our smoothing and probing scheme. The *critical bandwidth* in the future, at a given interval, is provided by its critical interval. To find the minimal bandwidth requirement for the critical interval, we apply the smoothing scheme, which spreads the constant sized packets evenly across the window. This leads to a sending difference between consecutive packets, which is defined by:

$$\triangle r = \frac{\triangle W}{pkts\_in\_critical\_interval}. \tag{2}$$

According to Keshav [22], the bottleneck bandwidth of a connection can be estimated by the packet pair approach at the receiver site. The essential idea is that the inter-packet spacing will be proportional to the time required for the bottleneck router to process the second packet. The *bottleneck bandwidth* is calculated as:

$$b = \frac{packet\_size}{\triangle r}. \tag{3}$$

Instead of using the average algorithm for the current interval, we use the bottleneck bandwidth to probe and obtain feedback before the network has to support the critical bandwidth.

In the third diagram of Figure 2, we present the PLUS probing scheme. To increase the chance of successfully transmitting critical packets, we only utilize B frames to probe the network. In each interval

$\triangle W$ we send B frames in intervals of $\triangle r$, while critical I and P frame packets are transmitted with the smoothed interval $\triangle a$. The idea is to force the network to give feedback on whether it will support our future bottleneck bandwidth. This allows us to adapt to the current situation *in advance* before we send the critical packets of the bottleneck bandwidth.

The advantage of PLUS is that B frames give feedback ahead of time if the critical bandwidth with critical packets can be supported. In case of loss of B-frames, the result is only a slight degradation of quality (in comparison to loss of a complete I to I sequence, which might contain 30 frames). A higher number of consecutive B packets of course increases the probing ability of the PLUS protocol. This can be controlled at the compression time of the stream. Since we only utilize B frames we just punctually increase the bandwidth to the same level as it will be required later by critical packets.

To be synchronized with the timing schedule of the media stream, we have to wait $\triangle i$ of time (Equation 4) after a sequence of probing packets,

$$\triangle i = nc * \triangle a - (nc - 1) * \triangle r. \tag{4}$$

where *nc* is the number of sequential non-critical B packets between critical I or P packets in window $\triangle W$.

This gap ($\triangle i$) serves as a bandwidth buffer, which further increases the survival rate of critical packets. In congested networks, routers tend to drop packets that arrive in bursts, and gaps in streams allow a busy router to process a different stream and accept later arrival of critical packets.

The key to a probing scheme is how multiple clients work together. In PLUS, probing consists of three phases (Figure 2). At the beginning we send critical packets with the average bandwidth (*averaging phase*), followed by the probing sequence (*probing phase*) of the B packets and a reduced bandwidth use (*resting phase*) before the next critical packet sequence starts. These three phases interact with each other, when multiple PLUS and non-PLUS streams are in use. We can identify the following scenarios to analyze the behavior of PLUS with multiple clients:

- *Single probing.* One PLUS stream probes the current network situation created by non-PLUS streams.
- *Sequential probing.* PLUS streams probe one after another. This will lead to an overlap of the probing phase of one stream with the averaging or resting phase of other PLUS streams.
- *Concurrent probing.* Multiple PLUS streams are concurrently in the probing phase, sending probe packets with the bottleneck bandwidth of the future.

The drawback of the current network protocols is that they lack the knowledge of future bandwidth requirements of a connection. To implement congestion control these protocols rely on the current network situation. The advantage of a single probing PLUS stream is that it has knowledge of the future bottleneck

requirements and reduces proactively its sending rate if the current network situation signals that it can not support it. Note, PLUS is a best effort approach. A sudden increase in network load right before the PLUS stream sends its critical interval can lead to loss of critical packets. But this approach is still more proactive in congestion control than currently implemented protocols.

In the sequential probing scenario, the PLUS probing sequences do not overlap. This is similar to the single probing scenario. In this case, a PLUS stream probes against averaging and resting phases of other PLUS streams and compares it against its critical interval. The estimated result might lead to loss in case multiple PLUS streams send their packets within the critical interval at exactly the same time, each assuming the network can support it. However, the probability of probing only against average and resting phases of other PLUS streams is negligible and can be further reduced by increasing the number of B frames in a stream [1].

Concurrent probing of PLUS streams is the ideal scenario. If concurrent probing occurs, the PLUS protocol reports a conservative estimation of the current network situation. The estimation is based on the bandwidth needs of the critical intervals. This behavior allows PLUS streams to protect critical packets when the maximal capacity of a connection is reached. If the concurrent probing causes packet loss, PLUS streams back off, harmonizing with TCP and other PLUS streams.

The advantage of our scheme is that adaptation based on feedback from probing gives the server, rather than the routers, control over packet dropping, which will increase the chance of successfully transmitting *critical packets.*

3.2 Back-off and Recovery Policy

If the system experiences packet losses during the probing scheme, it is an indication that the network can not support the critical bandwidth for the given stream. The system needs to back off and test what is the maximal bandwidth that the network can currently provide. As a back-off policy, any TCP-friendly adjustment scheme can be utilized. In the congested state, probing discontinues until the steady state is reached again.

In this section, we present an easy to use back-off and recovery policy for the PLUS scheme, which incorporates a high sensitivity towards packet loss as recommended by the IETF (Internet Engineering Task Force). For each smoothing interval, the PLUS protocol checks if all packets sent since the last check have arrived at the client site. If a packet is lost, the PLUS algorithm initiates a multiplicative back-off. If no further packet loss is detected, the system recovers additively for each consecutive interval $\triangle W$ until the maximum sending rate is reached again. The advantage of this approach is a fast reaction in time of a traffic jam and a conservative increase of bandwidth to avoid further congestion.

[1] The probability of this situation can also be reduced if the server provides a synchronous sending schedule for all streams with critical intervals. The new schedule is bound by the amount of prefetching at the client site, which allows modifications of the stream sending rates.

To be conservatively responsive to packet loss, we introduce the concept of *multiplicative adjustment factor* ω, which depends on the number of packets sent, $\overline{\#}$, and acknowledged, $\tilde{\#}$, within the interval $\triangle W$:

$$\omega = \frac{\overline{\#}}{\tilde{\#}}. \tag{5}$$

This leads to a new timing difference between packets:

$$\triangle r_{new} = \omega * \triangle r. \tag{6}$$

Thus, the percentage of data loss has direct influence on the adjustment factor ω, which will lead to a wider smoothing interval and reduction of required peak bandwidth. The amount of adjustment is dependent on the size of the interval $\triangle W$. A larger interval allows more packets to be sent, which will decrease the influence of short burst packet loss.

In the presence of no losses, the recovery policy increases the number of acknowledged packets of the congested interval by one for each upcoming interval $\triangle W$. This leads to a new ω:

$$\omega = \frac{\overline{\#}}{\tilde{\#}+1}. \tag{7}$$

Multiplying the new ω with $\triangle r$ leads to a decrease of the sending interval $\triangle r_{new}$. Equation 7 takes into account the possibility of loss of all packets. In this case we only send one packet per interval to gather feedback if the network has recovered. If the network has recovered, we increase the sending rate up to the original probing value $\triangle r$.

Table 2 gives an example of the back-off and recovery policy. For simplicity, we assume that the server sends 10 packets every interval. We also assume a negligible round trip time (changes are in effect during the next interval). By stretching the sending intervals we reduce the bandwidth in case of congestion. If the

	$\triangle W1$	$\triangle W2$	$\triangle W3$	$\triangle W4$	$\triangle W5$
Pkt_loss	-	2	-	-	-
$\triangle r_{new}$	$\triangle r$	$\frac{10}{8}\triangle r$	$\frac{10}{9}\triangle r$	$\frac{10}{10}\triangle r$	$\frac{10}{11}\triangle r$
$\triangle r_{send}$	$\triangle r$	$\frac{10}{8}\triangle r$	$\frac{10}{9}\triangle r$	$\triangle r$	$\triangle r$

Table 2: Back-off and recovery example.

number of packets per interval $\triangle W$ is small, the reduced bandwidth might be enough to support the timing schedule. This can be done because we normally send with the critical, not with the minimal bandwidth. On the other hand, the adjusted sending interval may be so large that it cannot support the schedule. The PLUS protocol detects this situation by performing the following check at the sending site:

$$\triangle r_{send} > \triangle a, \tag{8}$$

where $\triangle a$ represents the length of the sending interval required to send all packets in $\triangle W$. If (8) is true, then the system must adjust the schedule so that fewer media units are being sent within $\triangle W$. One way to do

this is to drop media units until $\triangle a$ becomes less than $\triangle r_{send}$. The back-off and recovery policy presented for the PLUS protocol is highly adaptive towards packet loss. Since it is not based on any reduction factors [35], it is highly stable and does not oscillate for higher packet loss rates.

To guarantee that no critical packets will be dropped, the PLUS protocol provides priority-based proactive packet dropping. Packet dropping of a leading I-frame in an I to I sequence for a MPEG encoded video stream will render the whole series useless. The PLUS protocol takes this into account by providing a priority based media unit dropping scheme. The idea is to proactively reduce the bandwidth to avoid further congestion, be TCP-friendly and increase the survival rate of critical packets.

I	P	B	B	P	B	B	I
0	1	2	3	4	5	6	

Table 3: MPEG sequence.

Table 3 provides an arbitrary I to I sequence. The P-frames can only be decoded with the help of the I-frames and the B-frames rely on the information encoded in the I and P-frames. The prioritized dropping scheme for video preprocesses the media stream and finds the distance between two adjunct I-frames (in our example the distance is 7). Then a drop order for these frames is generated: first, the B frames in order of rising sequence numbers will be dropped; and second, the P frames and last the I frame of the sequence. If Equation 8 is true, $\triangle r_{send}$ will become the new smoothing interval applied to all packets and frame dropping must be initiated to fit all packets within the timing interval $\triangle W$. Each frame dropped buys time in the form of the sending rate (1/30 sec for a rate of 30 frames per second) which allows a stretch of $\triangle r_{send}$ and a reduction of bandwidth. The dropping sequence generated for the example in Table 3 is: first the B-frames with numbers 2, 3, 5, 6, then the P-frames 1, 4 and last the I frame with number 0. Packet dropping of course influences the QoS of the presentation and can for multimedia presentations only be performed to a certain degree. The amount is bound by the QoS specification of the user (for example, a maximum drop rate of 50%). This in general is a drawback in TCP friendliness of all multimedia streaming protocols. Prefetching some amount of data can allow a higher drop rate in case of congestion (assuming the congestion lasts only a short period of time).

In case that packet dropping is sufficiently allowed, one method to analyze the behavior and TCP friendliness of the PLUS protocol is to compare it with a TCP bandwidth usage model. The relationship between throughput (T_{TCP}) and loss (p) for TCP was provided in [14, 40] as:

$$T_{TCP} = \frac{1.22 * MTU}{RTT * \sqrt{p}}, \tag{9}$$

where MTU is the maximum packet length and RTT is the round trip time of the connection. Note that this model is only partially correct [35] up to data loss of 16%. But it still gives us a feel for the TCP behavior in case of packet loss.

We performed the same analysis for the PLUS protocol. After one loss PLUS recovers in the time of $2 * \triangle W$, which leads to an average steady state throughput of a PLUS connection:

$$T_{PLUS} = \frac{(\overline{\#} - \frac{1}{2}) * pkt\_size}{\triangle W}. \tag{10}$$

Combined with the loss ratio of the PLUS connection

$$p = \frac{1}{2 * \overline{\#} - 1}, \tag{11}$$

we achieve a PLUS throughput estimation of:

$$T_{PLUS} = \frac{pkt\_size}{2 * \triangle W * p}. \tag{12}$$

Analyzing the differences between the PLUS and TCP throughput models in Formula 9, it shows that the loss influences the throughput of TCP connections by $1/\sqrt{p}$ in comparison to $1/p$ in case of PLUS. This means that PLUS also reacts proactively in reducing its load in case of congestion, gradually narrowing down to TCP behavior with increase in congestion. Thus, both TCP and PLUS losses consider the packet drop rate and are based on the amount of time it takes to recover from the loss of one packet. The packet drop rate controls the average steady state throughput or sending rate at the source. This relationship is given by Formulas 9 and 12. Note that Formula 12 overestimates the throughput of PLUS since the sending rate is bound by critical interval (i.e. the sending rate will not increase more than critical rate even there is an available bandwidth).

Prioritized packet dropping has the advantage of providing bandwidth reduction at the server site, allowing the server to choose which packets to eliminate instead of routers. This will increase drastically the provided QoS to the client.

4 Implementation and Experiments

In this section, we describe the performance of the PLUS protocol. We first introduce the platform on which the PLUS protocol is implemented. We designed and implemented a client-server distributed presentation system that can flexibly and adaptively support streaming of media data across the Internet with the help of the PLUS protocol. The system called *NetMedia* has four main components: *client* for presentation scheduling, *server* for resource management and scheduling, *database system* for data management and storage, and the *PLUS-protocol* for handling the data transfer over the network [44].

4.1 *NetMedia* System

Server design. The server design must support the PLUS protocol as well as access to individual streams while sharing the resources among all streams. A multimedia presentation may contain three types of

media streams, video, audio or transparencies. The tradeoff between individual and shared resources is addressed in our server design. Multiple threads are used in implementing the module components. The system manages an *Admitted Client Set*. Each admitted client has one buffer and one packet thread for each requested stream, and a probing thread. The packet thread reads media units from the buffer and cuts out packets of constant size to be delivered to the network. The probing thread receives acknowledgments from the client and updates the sending rate of each stream according to the PLUS protocol. There are also control messages which are used to start or end presentation playback. The *Disk Read Thread* is a shared resource which constantly browses over the *Admitted Client Set*, reading data from databases, and writing the data to the buffers of each admitted client. If a buffer of one admitted client is full, the Disk Read Thread does not block, but moves to the next buffer. The integration of the above design strategies provides highly efficient management of the media data retrieval, buffering and adaptive streaming at the server site.

Client design. The main feature of the client design is to provide a multimedia stream class that provides methods for synchronized retrieval and presentation of multimedia data to a user programmer. The PLUS protocol uses a TCP socket for acknowledgment of feedback messages like rate adjustment and congestion of network. A TCP connection is used to guarantee no loss of ACKs. In the client, a service provider, termed MultiMediaRealTimeStream, is implemented to support all the above described services. With the MultiMediaRealTimeStream service, an application can be written in C++ or Java to communicate with the server, get well behaved UDP media streams and display them through its own interface. When a MultiMediaRealTimeStream is opened, it first asks for admission, then it will start all related threads. Our design can effectively and efficiently integrate different servers, the network and the client to support flexible and dynamic multimedia data transfer service over the Internet without unfairly taking bandwidth from concurrent TCP connections.

Details on the *NetMedia* system can be found in [38, 44].

4.2 Test Environment

In order to effectively test the PLUS protocol, we established a video capture testbed capable of capturing lecture material provided at the University at Buffalo. The testbed consists of a video-camera, a SUN-VideoPLUS capture card, a SUN-Ultra 10 workstation with 128 MByte of memory and a 300Mhz RISC processor. The SUN VideoPlus capture card provides MJPEG, MPEG-1, H.261 and H.263 compression for 30 frames per second. Audio is supported with the following encoding standards: G7.11 (Alaw, μlaw) G.722,G.728,G.723.

Lectures captured for our system were encoded using the MPEG-1 compression technique. The I-P-B sequence could be varied. Streams were encoded with four P and 10 B-frames per sequence (I-BB-P-BB-P-BB-P-BB-P-BB). Audio was captured using μlaw encoding without any compression. Audio in our

experiments never provided a critical data size in comparison to video.

The server can be established on any SUN-OS compatible machine. To cover a wide variety of network situations, we established test sites in the local area, neighboring states and sites across both the Atlantic and Pacific oceans. We ran the server at our Buffalo campus, Turkey, Japan and Purdue University, while clients could connect from SUN Ultra 5 machines in Buffalo. For this study, test runs were taken every hour over the period between March 1999 and October 1999. All figures are averages over a week of measurements at the given time points. The server requires about 0.15% system resources on a SUN Ultra 10-300 with 128Mbyte, delivering audio, video and transparencies, and uses 2.2 Mbyte of disk space. For database access, an additional remote procedure call (RPC) server (277k) needs to be started. During the tests, the client resided in our lab at Buffalo. We used a Sun Ultra 5 with 64 MByte of memory and a SUN creator 3D graphic card, which supports 24bit color depth for video.

To compare the enhancements achieved by the PLUS algorithm, we implemented it on top of an existing end-to-end network delay control protocol, termed DSD (difference between sending and display) [38]. In general, DSD sends the data units comprising the media stream according to the display rate of the

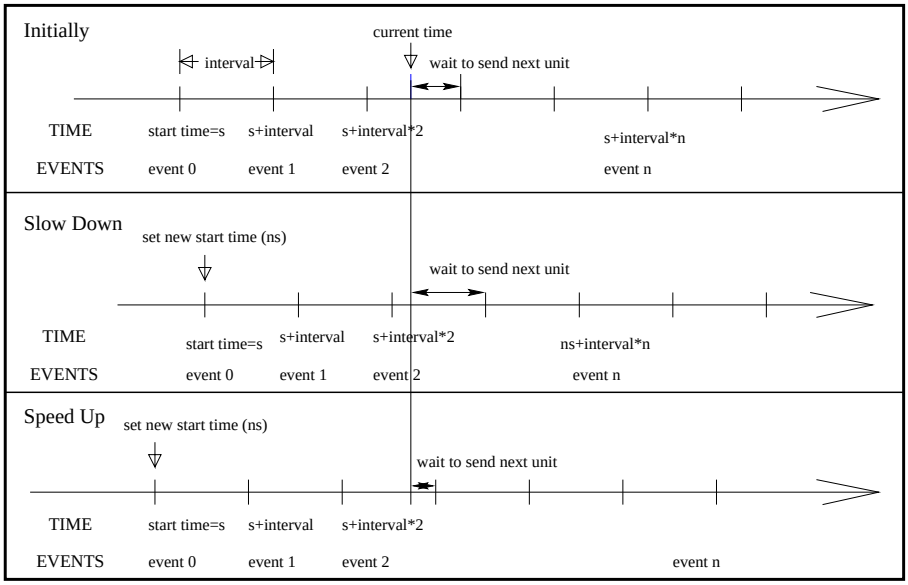

Figure 3: DSD-Adaptation.

presentation. However, because delay in the network may change, DSD will adjust the time difference between sending and display in order to avoid data loss due to buffer starvation or overflow at the client. At any calculation point, if the optimal time difference is determined to be different from the current time difference, then the client sends a feedback message to the server site to make an adjustment. One of the recent protocols, TEAR, also computes the sending rate at the receiver and sends the adjustment rate as feedback to the sender like DSD. Figure 3 illustrates the adjustment scheme of DSD, which is based on

recomputing a new presentation start time. In case of slow down, the start time is moved to the future (in relationship to the old start time), which results in a delay of the next data unit. In case of speed up, the start time is moved to the past, resulting in sending the next data unit sooner than the initial timing schedule.

The DSD scheme is well suited to provide synchronization between a client and a server. The advantage of this algorithm is that the DSD gives control over the current network delay, allowing control over the buffer levels at the client site and thereby avoiding data loss to the presentation. The drawback is that the DSD algorithm lacks responsiveness towards packet loss. The adaptation is only triggered by the delay of packets that reach the client. Also, the speed-up/slow-down adjustment only effects the delay and does not reduce the bandwidth for a longer period of time, thereby suppressing other TCP connections in case of congestion.

The PLUS protocol is an ideal addition to an end-to-end scheme like DSD. It adds packet loss awareness and bandwidth adjustment to DSD, with the idea of protecting critical packets for multimedia presentations.

4.3 Experimental Results

Experiments for the PLUS protocol were conducted within the *NetMedia* system. We compared the PLUS protocol on top of DSD with a smoothed version, using 5 seconds smoothing intervals. All protocols are based on UDP. DSD delivers the frames according to the time schedule of the stream. Packets in each frame are sent at maximum network speed. To reduce burstiness of streams, smoothing techniques can be applied. The five seconds smoothing approach uniformly spreads all packets belonging to frames displayed within five seconds (averaging bandwidth scheme). Playback time can start as soon as enough prefetch data has arrived. The PLUS protocol also uses a 5 second smoothing window for I and P frames. B frames are sent with the minimum smoothing interval ahead, leaving mostly a gap between the last frame in a series of B frames and the first packet of a *critical* I or P packet.

The simplest measure of network performance is the average packet loss. A packet is considered to be lost if it is not received by the client or received too late to be used for display. If the difference between the sequence number of a new arriving packet and that of last arrived packet is larger than 1, the packet is also considered as data loss rather than out-of-order packet by PLUS to react faster to network congestion. The average loss rates for all paths are shown in Table 4.

On Campus	Purdue	Japan	Turkey
0.01%	1.3%	7.53%	53%

Table 4: Average packet loss.

Network connections at the Buffalo Campus can be considered nearly 100% loss free. All connections run at 100baseT supported by a FDDI backbone passing through at maximum of one router. The network

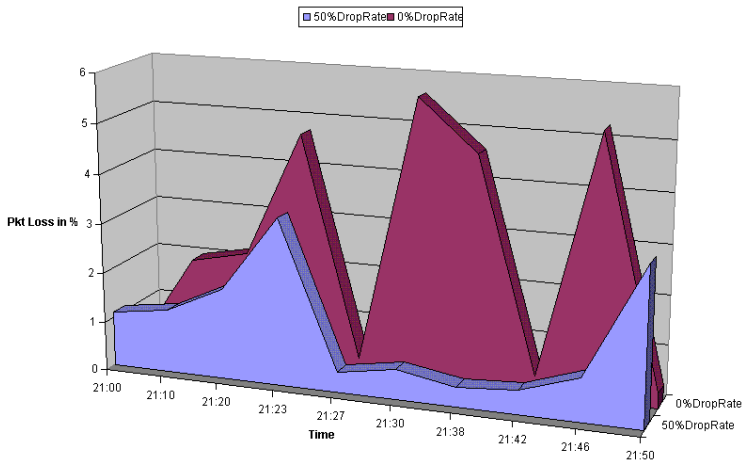

Figure 4: Effect of proactive frame dropping on packet loss with server in Turkey.

load experienced during the experiments is classified as *ideal*. Experiments at Purdue and Japan experience *light* packet loss during most of the time. Packets normally traveled through 11 (Purdue) and 24 (Japan) routers. Test runs at Turkey produced *heavy* traffic most of the time. Even though the average loss was 53%, runs during prime time of the day could easily produce results with loss rates up to 90%, which renders any presentation useless. The Turkey connection normally passes 15 routers. Due to its high traffic load, the Turkey account provided the best test-environment for congestion control algorithms. Since the state of the network changes often, average readings presented over time as in Table 4 are not as useful as average instantaneous readings. Therefore, experiments are conducted at specific times and the averages of these instantaneous readings are used to check the effectiveness of the PLUS protocol. The term "average" reflects the average instantaneous readings through the rest of the paper unless specified otherwise.

To get a clear picture of the loss reduction achieved by proactively dropping frames at the server site, we conducted test runs with the original stream and a preprocessed stream. The preprocessed stream drops 50% of its frames, starting first with B frames, then P frames and last I frames at the server site before the data is sent out. As an example, consider a stream consisting of 41 I-frames, 81 P-frames and 1088 B-frames. A 50% drop rate would eliminate 605 B frames uniformly over the playback time of the stream.

Figure 4 shows the average loss rate history between Buffalo and Turkey. In general, the reduced stream produced lower packet loss rates. The reduced stream eliminates non-critical packets while decreasing the loss rate on critical packets. The packets of the original stream are dropped by the network randomly. Adaptively reducing the sending rate will definitely help a stream to deliver its content with a lower packet loss rate on critical packets in case of congestion.

Smoothing techniques spread packets uniformly to reduce the bandwidth requirements and burstiness

of a stream. It has been proved that smoothing algorithms have an impact on the network behavior ([37]). We compared PLUS with DSD against a 5 second smoothing implementation. Experiments are conducted for each specific time (as specified in $X - axis$ of Figure 5) during the test period through Turkey. In the experiments, the PLUS protocol did not use the back-off policy explained in Section 3.2. The results (Figure 5) indicate that probing with proactive packet dropping decreases the loss ratio during congestion.

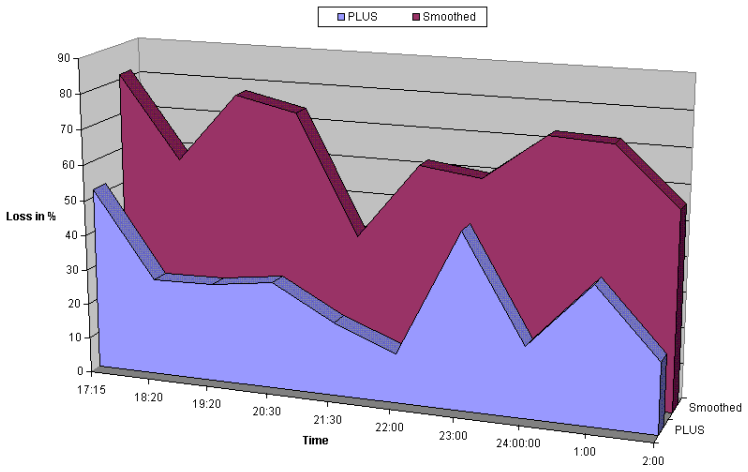

Figure 5: Comparison of the PLUS protocol against 5 sec smoothing without back-off with server in Turkey.

The PLUS protocol works best with an increasing amount of traffic because the combination of probing and backing-off in case of congestion provides the most chance of success in an environment, in which the router has the choice of punishing non-well behaved streams.

To test the PLUS protocol under *light* conditions we also ran several tests at Purdue and Japan. We compared the PLUS on top of DSD with DSD alone. Figures 6 and 7 present the average packet loss of the experiments conducted at specific times during the test period. Due to the fact that DSD uses the maximum sending rate for each packet belonging to one frame, it is more prone to loss of *critical* packets. Even though the load to Purdue is very light, we can see a slight advantage of the PLUS protocol in comparison to DSD and the smoothed version. In test runs to Japan, we experienced on average lower packet loss with PLUS in comparison to DSD. This can also be explained by the higher sending rate of DSD for packets belonging to the same frame, causing routers to more likely drop packets.

The PLUS protocol tries to avoid the random frame dropping by routers. This is achieved by probing the network and proactively dropping media units at the server site. To test the effect of proactively probing, we compared the number of received and successfully decoded frames (Figure 8) between the smoothed and the PLUS schemes. Loss rates do not necessarily provide a true indication of the quality of service. If mostly critical packets are effected, then the viewing experience is very low, even though the throughput is

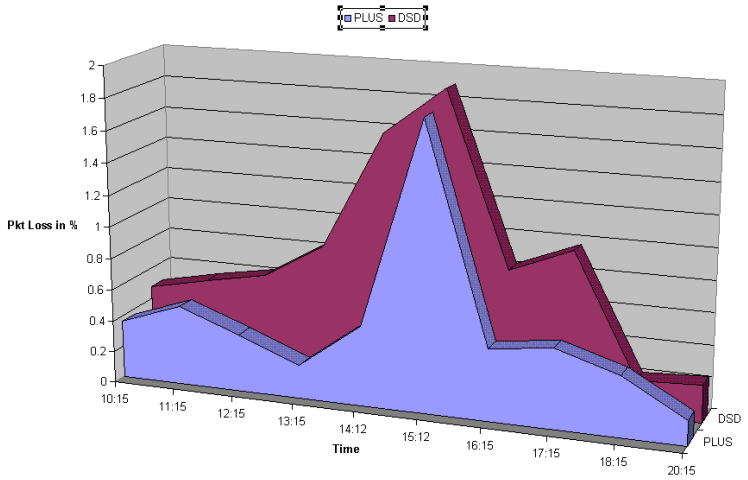

Figure 6: Comparison of DSD and PLUS on top of DSD with server in Purdue.

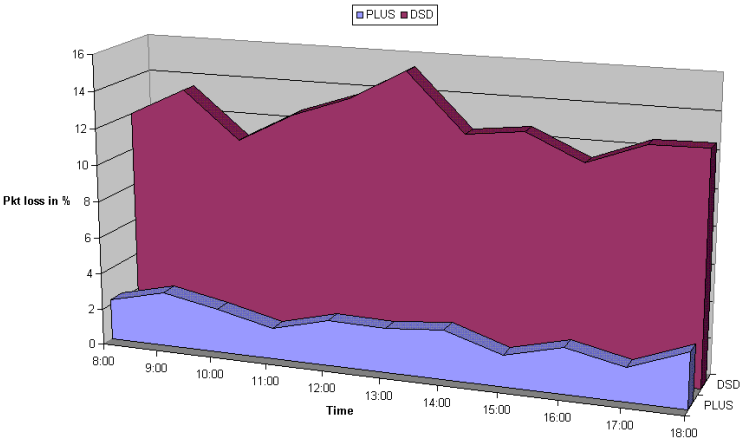

Figure 7: Comparison of DSD and PLUS on top of DSD with server in Japan.

high. The number of successfully displayed frames gives a truer indication of the QoS of the presentation.

The PLUS scheme on average could deliver 47% more frames to the client than the smoothed version. This is achieved by backing off the sending rate in detection of congestion. To study the effect on proactively testing the network we compared the PLUS protocol against a smoothed version, which backs off if the packet loss exceeds 10% within a 5 second test interval. As can be seen in Figure 8, the PLUS protocol does not just rely on the back-off mechanism to protect its content. The increased number of saved B frames (even though they are sent more aggressively) could be achieved because a larger number of *critical* frames has been saved. This is obtained by the probing nature of the PLUS protocol. Transmitting B frames with a higher bandwidth causes the router to more likely to drop this kind of frame (and initiate proactive frame dropping). Probing also pauses before a *critical* packet stream (thereby giving the router a chance

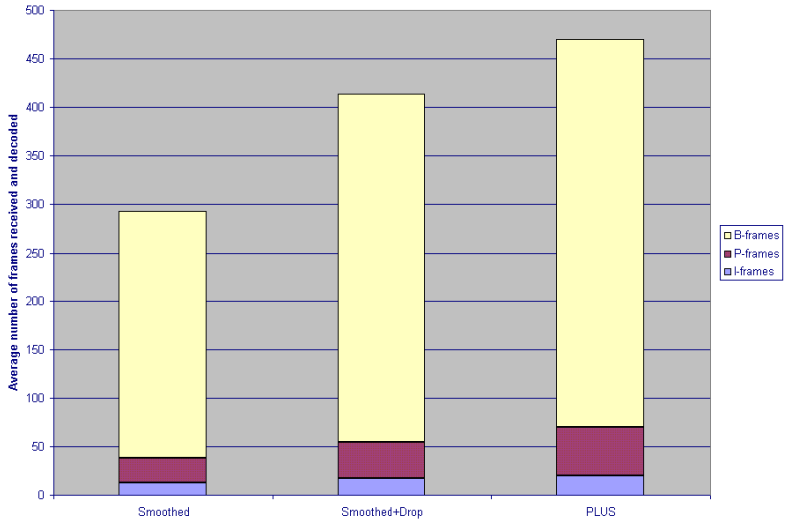

Figure 8: Comparison of received and decoded frames with server in Turkey.

to switch to a different stream and process the *critical* packets later), which increases the survival rate of *critical* packets. In Figure 8, the number of I-frames that are received and decoded when PLUS protocol is used is more than when smoothing with or without drop is used. Since the number of I-frames that are decoded increased, the number of P-frames that can be decoded will also increase. The number of B-frames that can be decoded will increase since the number of I and P frames that are decoded increased. In our test environment, decoding of one I-frame enables the decoding of 4 P-frames and 10 B-frames. The decoding of a P-frame also enables the decoding of one P-frame and two B-frames. Since the PLUS protocol increases the survival rate of critical packets, it increased the number of frames that can be decoded. Under light congestion, PLUS delivers slightly more usable data than smoothing or streaming over DSD. Figures 9 and 10 compare the received and decoded number of frames to the displayer with server in Purdue and Japan, respectively.

We also measured out of order packet arrival at the client site. In our experiments to Purdue, Japan and Turkey the routers rarely switched routes for packets. This resulted in a very small number of out of order packet arrivals at the client. In the experiments to Japan, out-of order packet arrival was more common.

In our implementation, we consider out of order packet arrival as a sign of potential congestion. The difference in packet loss between the DSD only and PLUS on top of DSD can be explained by the faster reaction of PLUS towards missing packets. DSD is well aware of the packets arriving at the client and adjusts accordingly to provide a stable data delivery to the displayer, but out of order packets and data losses do not influence its decision making. The PLUS protocol is an ideal complimentary tool to increase the amount of successfully delivered media packets.

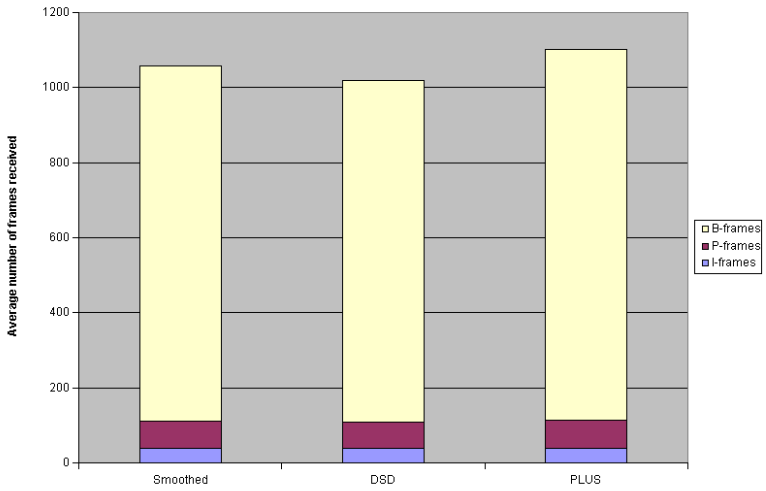

Figure 9: Comparison of received and decoded frames with server in Purdue.

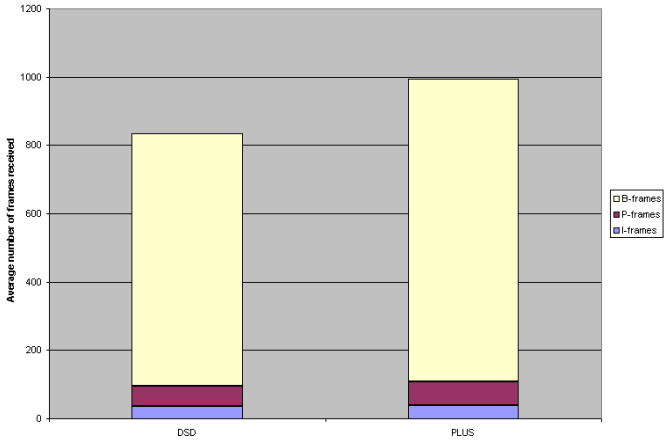

Figure 10: Comparison of received and decoded frames with server in Japan.

Figure 11 illustrates the number of out of order packets received from the DSD in comparison to the DSD combined with PLUS. The sending rate for packets in DSD, which are sent within a frame interval, increased the chance of out-of-order packets. Out-of order packets face the risk of packet loss due to timing requirements. An alternate route decided by a router may increase the end-to-end delay designed for this packet and renders it useless for presentation. This explains the higher packet loss experienced by the DSD only protocol in Figure 7.

As expected the smoothed version can improve the DSD only version due to its uniformly spread packets, which does not expose *critical* packets to a higher bandwidth. Even though the combination of PLUS and DSD hardly backed off during the test runs (due to the light load), it still performed the best in saving *critical* packets.

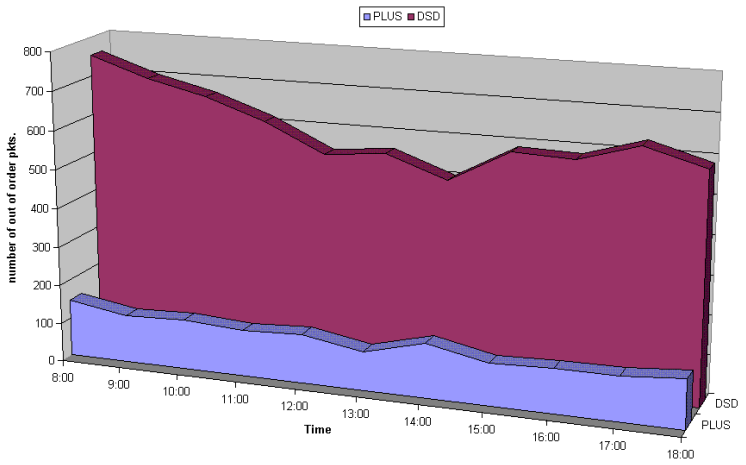

Figure 11: Comparison of out of order packet arrival with server in Japan.

The smoothing window used by PLUS protocol introduces delay between transmission and playout. This delay may be significant for applications requiring synchronization. Jitter is an important measure whether this delay is significant or not. The jitter experienced during the test runs of two presentations (CSE530 is a lecture presentation and XZEIN is a randomly selected presentation) are shown in Figures 12. Measured using the synchronization algorithm given in [21], the jitter obtained on Buffalo Campus remains on average in the ranges of 0-60 milliseconds (ms) throughout the presentation, which is much less than the upper bound (80 ms) given in [39]. Jitter obtained from Purdue, Japan and Germany experiences spikes in case of packet loss. On average the jitter is still below the upper bound 80 ms. Jitter to Turkey is influenced by the high traffic load experienced most of the time during transmission. Loss rates of 80% in a 5 second interval produce the jitter spikes. In case of reduced network load, the measured jitter also performs under the upper bound 80 ms. These experiments show that the PLUS protocol can effectively be used even when applications require fine synchronization.

4.3.1 Discussion

The end-to-end flow control scheme DSD provides just in time delivery, dynamic adaptive behavior to the network situation, and is successful in increasing QoS for the viewing experience. However, it does not address the issue of congestion control and TCP-friendliness, nor does it utilize the knowledge of its streams to improve transmission.

The PLUS protocol is used to increase the survival rate of critical packets in case of congestion. To do this it provides a probing mechanism, which utilizes the knowledge of the future bottleneck bandwidth in the delivered streams. The PLUS protocol is TCP-friendly, scaling down its bandwidth requirements in case of packet loss. This allows neighboring TCP connections to recover and prevents their starvation.

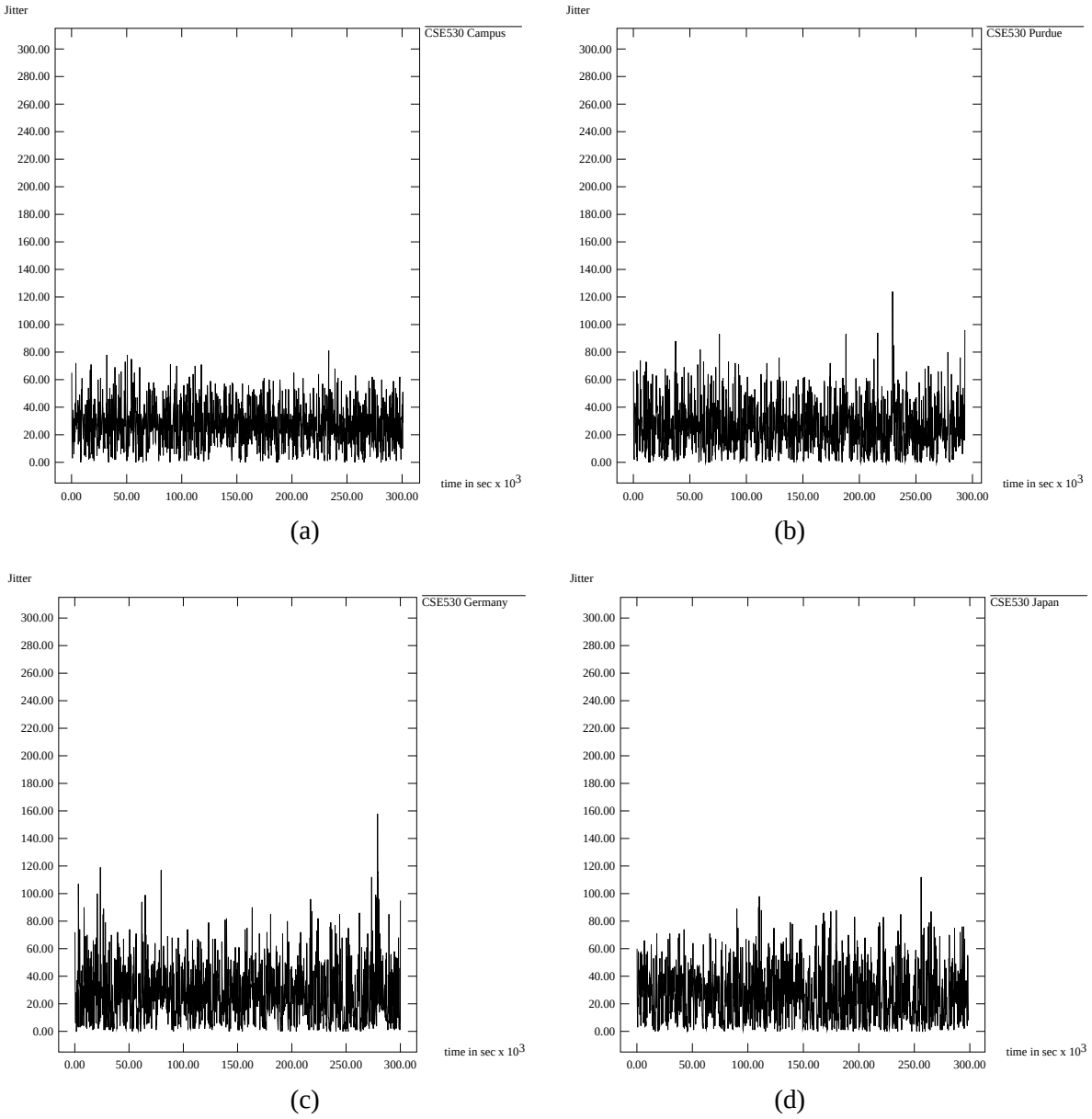

Figure 12: Jitter between audio and video: server at (a) Buffalo Campus, (b) Purdue, (c) Germany, and (d) Japan.

Experiments showed the increase of received and decoded frames with servers in Turkey, Japan and Purdue. The increase is achieved by the probing nature of PLUS, utilizing knowledge of the network situation to initiate proactive frame dropping in case the network can not handle the bottleneck bandwidth. Proactive frame dropping increased the survival rate for critical packets, because their packets could be spread over a longer period of time, which is equal to a bandwidth reduction. The survival rate is also increased by providing a 5 sec smoothing window, which reduces the bursty nature caused by using the MPEG-1 compression scheme. An interesting fact is that the sending pattern also influenced the number of out-of-order packets and packet loss. The probing scheme, which consists of a sequence of averaging, probing and resting phases, increased the survival rate of critical packets. The resting phases in streams allow a busy router to process a different stream and increase the likelihood at successful transmission of critical packets. Since routers tend to drop packets in bursts, this interruption in the processing sequence increases the survival rate of PLUS streams.

5 Conclusion

We have presented a new flow and congestion control scheme, termed PLUS (Probe-Loss Utilization Streaming protocol), for distributed multimedia presentation systems. This scheme uses the probing of the network status to effectively react to data loss and prevent potential network congestion. The novelty of this scheme is that feedback on the available network bandwidth is collected before the data with high bandwidth requirements is sent. This has the advantage of *avoiding* congestion rather than only *reacting* to it after it happens. By using B frames as probing packets, no additional data overhead is generated, and I and P frames are better protected. As a result, data losses in the network are reduced. Experiments conducted using the *NetMedia* prototype support these conclusions.

References

[1] G. Abowd, C. Atkenson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney, and M. Tani. Teaching and learning as multimedia authoring: The classroom 2000 project. In *Proceedings of ACM Multimedia 96*, pages 187–198, Boston MA USA, 1996.

[2] D. Bansal and H. Balakrishnan. Tcp-friendly congestion control for real-time streaming applications. Technical Report MIT-LCS-TR-806, M.I.T. Labaratory for Computer Science, NCSU, May 2000.

[3] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *SIGCOM Symposium on Communications Architectures and Protocols*, pages 58–67, UK London, Aug 1994.

[4] Braden, Clark, Crowcroft, Davie, Deering, Estrin, Floyd, Jacobson, Minshall, Partridge, Peterson, Ramakrishnan, Shenker, Wroclawski, and Zhang. Recommendations on queue management and congestion avoidance in the internet. Technical Report draft-irtf-e2e-queue-mgt-00.txt, Internet Draft, 1997.

[5] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS Control of Multimedia Applications based on RTP. *Computer Communications*, 19:49–58, January 1996.

[6] K. Selcuk Candan, B. Prabhakaran, and V. Subrahmanian. Chimp: A framework for supporting multimedia document authoring and presentation. In *Proceedings of ACM Multimedia 96*, pages 329–340, Boston, 1996.

[7] S. Cen, C. Pu, and J. Walpole. Flow and Congestion Control for Internet Streaming Applications. In *Proceedings of Multimedia Computing and Networking (MMCN98)*, 1998.

[8] D.D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *SIGCOMM*, pages 106–114, 1988.

[9] D.D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: architecture and mechanism. In *SIGCOMM*, pages 14–26, 1992.

[10] W. Feng. *Buffering Techniques for Delivery of Compressed Video in Video-on-Demand Systems*. Kluwer Academic Publishers, 1997.

[11] W. Feng, F. Jahanian, and S. Sechrest. Providing VCR Functionality in a Constant Quality Video On-Demand Transportation Service. In *IEEE Multimedia 1996*, pages 127–135, Hiroshima, Japan, June 1996.

[12] W. Feng and W.Feng. The impact of active queue management on multimedia congestion control. In *International Conference on Computer Communications and Networks*, pages 214–218, Lafayette, 1998.

[13] S. Floyd. TCP and Explicit Congestion Notification. *Computer Comunication Review*, 24(5):10–23, Oct. 1994.

[14] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 1999.

[15] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993.

[16] S. Floyd and F. Kevin. Router mechanism to support end-to-end congestion control. Technical report, Technical Report, February 1997.

[17] E. Fox and L. Kieffer. Multimedia Curricula, Courses, and Knowledge Modules. *ACM Computing Surveys*, 27(4):549–551, December 1995.

[18] B. Haskell, A. Puri, and A. Netravaldi. *Digital Video: An Introduction to MPEG2*. Chapman and Hall, New York, 1996.

[19] V. Ozdemir I. Rhee and Y. Yi. Tear: Tcp emulation at receivers - flow control for multimedia streaming. Technical report, Department of Computer Science, NCSU, April 2000.

[20] Robert Denda J. Widmer and Martin Mauve. A survey on tcp-friendly congestion control. *IEEE Network Magazine*, 15(3), May 2001.

[21] T.V. Johnson and A. Zhang. Dynamic Playout Scheduling Algorithms for Continuous Multimedia Streams. *ACM Multimedia Systems*, 7(4):312–325, July 1999.

[22] S. Keshav. A control-theoretic approach to flow control. In *Proceedings of the Conference on Communications Architecture and Protocols*, pages 3–15, Zuerich, Switzerland, September 1991.

[23] F.L. Kitson, T. Malzbender, and V. Bhaskaran. Opportunities for Visual Computing in Healthcare. *IEEE Multimedia*, 4(2):46–57, 1997.

[24] I. Kouvelas, Hardman, and A. Watson. Lip synchronization for use over the internet: Analysis and implementation. In *GLOBECOM*, November 1996.

[25] L. Breslau and E. Knightly and S. Shenker and I. Stoica and H. Zhang. Endpoint Admission Control: Architectural Issues and Performance. In *ACM SIGCOMM*, September 2000.

[26] D. Lin and R. Morris. Dynamics of random early detection. In *SIGCOMM*, pages 127–137, 1997.

[27] J. Mahdavi and S. Floyd. Tcp-friendly unicast rate-based flow control. *Technical note*, 1997.

[28] S. McCanne and V.Jacobson. Vic: a flexible framework for packet video. In *Proceedings of ACM Multimedia*, pages 511–522, November 1995.

[29] G. Ozsoyoglu, V. Hakkoymaz, and J. Kraft. Automating the Assembly of Presentations from Multimedia Databases. In *Proceedings of the 12th Intl. Conf. on Data Engineering*, pages 593–601, New Orleans, Louisiana, February 1996.

[30] M. Parris, K. Jeffay, and F. Smith. Lightweight active router-queue management for multimedia networking. *SPIE Proceedings Series-Society for Optical Engineering*, 3654:162–174, 1999.

[31] M. Handley R. Rejaie and D. Estrin. Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proc. of IEEE INFOCOM*, March 1999.

[32] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ipv6 and to tcp. Technical Report draft-kksjf-ecn-00.txt, Internet Draft, 1997.

[33] K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transaction on Computer Systems*, 8(2), 1990.

[34] H. Schulzrinne, S. Casner, R.Frederick, and V. Jacobson. Rtp: a transport protocol for real-time applications. Technical Report RFC1889, Internet Engineering Task Force, January 1996.

[35] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A tcp-friendly adaptation scheme. In *NOSSDAV 98*, UK Cambridge, July 1998.

[36] D. Sisalem and A. Wolisz. Lda+ tcp-friendly adaptation: a measurement and comparison study. In *Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2000.

[37] S.Lam, S.Chow, and D.Yau. An algorithm for lossless smoothing of mpeg video. In *Conference on Communication and Architecture SIGCOMM*, pages 281–293, 1994.

[38] Y. Song, M. Mielke, and A. Zhang. NetMedia: Synchronized Streaming of Multimedia Presentations in Distributed Environments. In *IEEE International Conference on Multimedia Computing and Systems*, pages 585–590, Italy, Florence, June 1999.

[39] R. Steinmetz and K. Nahrstedt. *Multimedia: Computing, Communications and Applications.* Prentice Hall, 1995.

[40] T.Ott, J. Kempermann, and M. Mathis. Window size behaviour in tcp/ip with congestion avoidance algorithm. In *IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems*, June 1997.

[41] V. Elek and G. Karlsson and R. Ronngren. Admission Control based on End-toend Measurement. In *INFOCOM*, March 2000.

[42] B. P. Woolf. Intelligent Multimedia Tutoring Systems. *Communications of the ACM*, 39(4):30–31, April 1996.

[43] Yang Y.H., Dudoit S., Luu P. and Speed T. P. Normalization for cDNA Microarray Data. In *Proceedings of SPIE BiOS 2001*, San Jose, California, January 2001.

[44] A. Zhang, Y. Song, and M. Mielke. *NetMedia*: A Middleware Design Strategy for Streaming Multimedia Presentations in Distributed Environments. *IEEE Multimedia*. to appear.

[45] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, pages 8–18, September 1993.