Fairness of Adaptive Multimedia Applications[*][†]

Dorgham Sisalem
GMD-Fokus
Kaiserin-Augusta-Allee 31
10589 Berlin
Germany
Tel.: ++49 30 34 63 71 70
Fax.: ++49 30 34 63 81 70
sisalem@fokus.gmd.de

Technical Subject Area: Multimedia Communications

Abstract

In this study, we present a new scalable scheme for adapting the transmission rate of multimedia applications to the congestion level of the network. The scheme is based on the end-to-end Real Time transport protocol (RTP). Results achieved through simulations and measurements suggest the efficiency of the scheme in utilizing the network resources and decreasing the loss rates. However, with no support from the network, adaptive applications tend to be unfair towards long distance connections and lead to the starvation of competing TCP connections.

[*]This work was funded in part by the BMBF (German Ministry of Education and Research) and the DFN (German Research Network).

[†]In case of acceptance the paper will be presented by Dorgham Sisalem

1 Introduction

The vast majority of multimedia applications used currently in the Internet, such as VIC [14], VAT [13], NEVOT [2] and NEVIT [22] are based on the UDP and RTP transport protocols. However, both UDP and RTP offer no means of quality of service control and can not thereby guarantee any level of QoS. Due to the fluctuations in the network conditions, the inability of those protocols to support QoS control renders multimedia applications, often, useless. Two main approaches were introduced in the past to overcome this problem: resource reservation [3] and application control [7]. Reserving enough resources for supporting a certain QoS level in advance guarantees this quality level and would be the most straight forward approach for handling the problem. However, as it is usually impossible to know the exact characteristics of a certain stream in advance one would tend to over reserve resources to guarantee the requested QoS level. This would, however, lead to under utilizing the network and would get costly if one has to pay for the reserved but unused resources.

By trading off temporal and spatial quality to available bandwidth or manipulating the playout time of continuous media in response to variations in delay, multimedia flows can keep an acceptable QoS level at the end systems without necessarily requiring any changes in the network [5]. However, such an approach requires the exchange of state information between the source end system and the network as it is the case for the available bit rate service of ATM [17] or between the sender and the receivers as its is the case for TCP [23] and the scheme presented here. As these state information have to traverse at least a part of the network, the source will not be able to change its sending rate immediately after a change in the congestion state of the network.

In Sec. 2, we present a new scheme called the loss based adjustment algorithm (LBA) for adapting the transmission rate of multimedia applications to the congestion level of the network. The LBA scheme is an enhancement of a scheme that we previously presented in [18]. In Sec. 3, we test the performance of the scheme in terms of bandwidth utilization. Fairness aspects and interoperability of the scheme with TCP are presented in Sec. 4 and Sec. 5.

2 The Loss Based Adjustment Algorithm (LBA)

The loss based adjustment algorithm is based upon the Real Time transport Protocol (RTP) [16] designed within the Internet Engineering Task Force (IETF). Note that RTP is not a transport protocol in the common sense. That is, it offers no reliability mechanisms, has no notion of a connection and is usually implemented as part of the application and not the operating system kernel. 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 as seen by other members of the session.

RTP sessions consist of two lower-layer data streams, namely a data stream for, say, audio or video and a stream of control packets (RTCP).

Using the control protocol (RTCP), each session member periodically multicasts control packets to all other session members. The control packets contain information about the received and transmitted data rates, delay jitter and packet losses. The desire for up-to-date control information has to be balanced against the desire to limit control traffic to a small percentage of data traffic even with sessions consisting of several hundred members. Therefore, the control traffic is scaled with the data traffic so that it makes up a certain percentage of the data rate (usually 5%). This design implies that the number of RTCP packets sent by all members of a session should stay constant for sessions of different sizes. However, as the minimum interval between two RTCP packets is limited to 5 seconds, it is possible that a certain number of members does not consume all the bandwidth available for RTCP. Hence, increasing the number of members results in an increase in the number of RTCP packets sent. Actually, the scaling schemes of RTCP start showing effect only for sessions with a higher number of members than a scaling threshold (th_{scale}) approximately specified as follows:

$$th_{\text{scale}} = \frac{\text{min. Interval} \times B_{\text{RTCP}}}{\text{RTCP Packet size}} \tag{1}$$

with the minimum interval set to 5 seconds and B_{RTCP} as the bandwidth dedicated for RTCP. Detailed analysis of this issue can be found in [8].

During a conferencing session each receiver reports in its control packets the percentage of lost data noticed since sending the last control packet. At the sender site, the RTCP packets are processed and depending on the loss values reported within the RTCP packets, the sender can increase, decrease or keep its current sending rate, see Figure 1.

The decision of how to change the sending rate is taken according to the following rules:

- A source starts sending data with a pre-defined rate
- Initially, the sender is in the non-congested state. During this state the sender increases its transmission rate additively with a pre-defined additive increase

factor (AIF) divided by $(\min\,[\text{members}, th_{\text{scale}}])$ for each received RTCP message indicating losses below a pre-defined loss threshold (loss_{th}). The number of members is determined through the RTCP protocol and the scaling threshold (th_{scale}) can be calculated as in Eqn. 1.

$$\text{rate} \;\; = \text{sending rate} + \frac{\text{AIF}}{m} \tag{2}$$

- When receiving a control message with loss ratio $\geq \text{loss}_{th}$, the sender goes into the congested state and the sending rate is reduced as follows:

  $$\text{rate} = \text{sending rate} * (1 - \text{loss ratio} + \text{loss}_{th}) \tag{3}$$

  the identity of the member reporting this loss is noted as the *losing member* and the reported loss value is saved as *highest loss*.

- After entering the congested state, the sender waits for one of the following cases:

  – Until another RTCP packet from the receiver, who reported the loss is received. Depending on the loss report of the new RTCP packet the sender either goes into the non-congested state or reenters the congested state.

  – Control messages received from members other than the *losing member* with loss values higher than the *highest loss* lead to an additional rate reduction as follows:

  $$\text{rate} = \text{rate} * (1 - \text{loss ratio} + \text{highest loss}) \tag{4}$$

highest loss is then set to this new loss value and the *losing member* is set to the identity of the member reporting this new loss. In this case, the sender reenters the congested state.

– If neither of the above cases is valid, all RTCP packets are ignored for five seconds, to allow for the reduction to take effect. This time period is approximately the minimum interval between the sending of two consecutive RTCP packets.

Additionally, the user can specify a minimum rate threshold below which the sender should not reduce its rate.

To prevent rapid changes in the sending rate and accommodate temporal loss peaks the loss ratios used for determining the congestion state and the sending rate should be determined as smoothed ratios and not the actual loss ratios reported in the RTCP packets. That is, for each receiver i the sender should calculate a smoothed loss ratio $\text{loss}_{\text{smoothed\_i}}$ as follows:

$$\text{loss}_{\text{smoothed\_i}} = (1-\alpha) * \text{loss}_{\text{smoothed\_i}} + \alpha * \text{loss}_i$$

with loss_i as the loss value reported in the last received RTCP message from member i and α is a smoothing factor set to of 0.5. This value was chosen based on the different simulation studies run for the LBA algorithm [18].

3 Measurement of the Performance of the LBA Algorithm

To verify the performance of the LBA algorithm we ran a simple experiment over our local ATM network. As Fig. 2 depicts, we connected two workstations over a

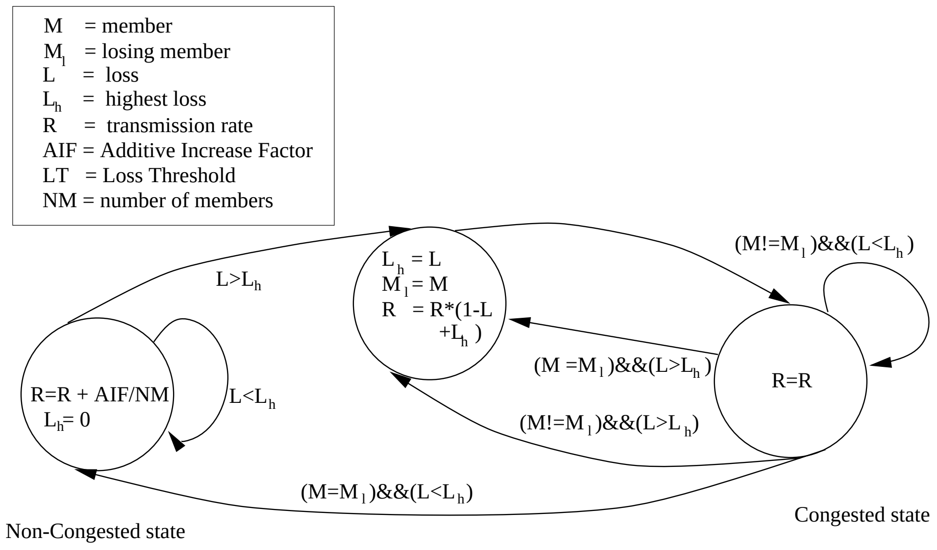


Figure 1: State diagram of the LBA Algorithm

permanent virtual circuit (PVC) with a limited capacity of 1 Mb/s. From one of the stations we sent two JPEG video streams using the NEVIT video tool [22] which we enhanced with the LBA algorithm. One of the streams, namely stream 1, had a maximum bandwidth of 1.5 Mb/s. The other stream was restricted to 0.5 Mb/s maximally. We have used an additive increase factor of 50 kbits/sec.

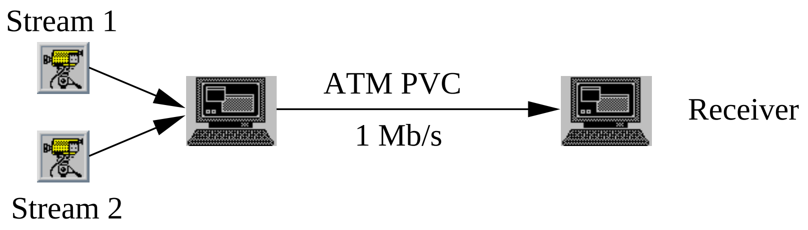
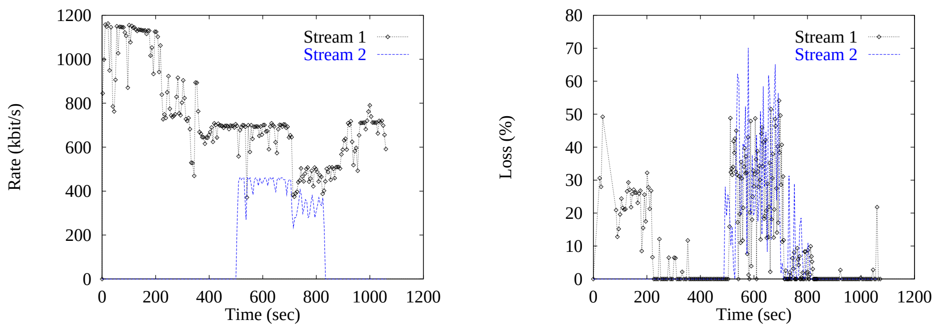


Figure 2: Measurement topology for the performance evaluation of the LBA algorithm

Fig. 3(a) shows the bandwidth produced by the two streams over a time period of 20 minutes. Fig. 3(b) shows the corresponding losses noted for each stream



(a) Bandwidth (b) Loss

Figure 3: Performance of the LBA algorithm

during the same time period. To show the effects of using the LBA algorithm we divided the experiment into different phases:

1. In the first phase, between the time 0 seconds and 200 seconds, the first video stream is sent with a rate higher than the maximal capacity of the PVC. This results in high losses as can be seen from Fig. 3(b).

2. In the time period between time 200 seconds and 350 seconds the LBA algorithm was applied resulting in a rate and loss reduction. As the algorithm tries to increase the used bandwidth, we still see some loss peaks whenever the video stream consumes more bandwidth than available on the PVC.

3. In the third period, from the time point 350 seconds until the time point 500 seconds, we stopped using the LBA algorithm. As the video stream was using a constant bandwidth below the capacity of the PVC no losses were noticed during this period.

4. In the next phase, between the time points 500 and 650 seconds, a second video stream was sent on the same PVC. As the sum of the bandwidth used by both streams exceeds the PVC capacity, Fig. 3(b) shows high losses during this period for both streams.

5. Up from the time point 650 seconds we switched the LBA algorithm on again. We can see that both streams share the PVC bandwidth more or less fairly, and both have rather low losses.

6. Finally, after switching the second stream off at the time point 850 seconds, we notice that the first stream increases its transmission rate.

Performance tests based on simulating the scheme with different patterns of bursty background traffic and a look at the scalability of the scheme to multicast sessions with different numbers of members can be found in [18]. That work shows that this scheme can easily accommodate different traffic patterns and scale with groups ranging from 2 members up to several hundreds of members without any performance degradation.

4 Fairness of the Loss Based Adjustment Algorithm

Adaptive applications are from a certain point of view similar to the available bit rate (ABR) service proposed for ATM. Both schemes adapt the sending rate of the end systems to the network congestion state by increasing the sending rate during underload situations and decreasing it during congestion periods. However, while in the ABR case the network determines the appropriate rate the senders should use, in our case the senders determine the sending rate using the control messages sent by the receivers.

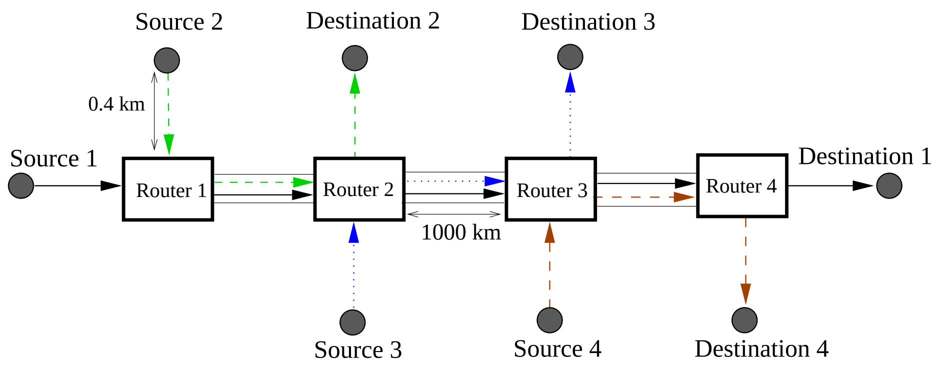
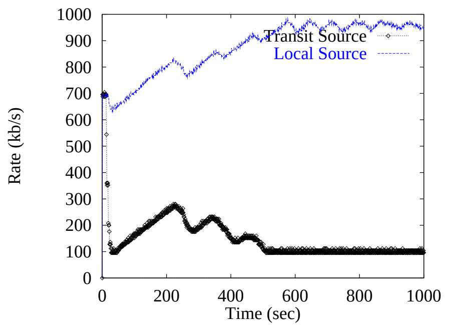


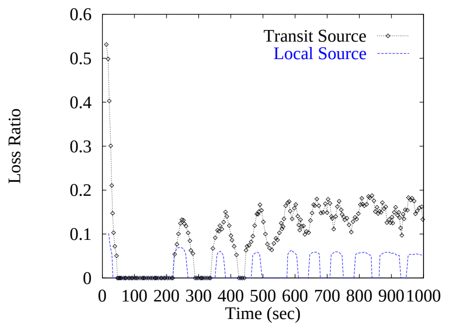
Figure 4: Network topology for investigating the max-min fairness behavior of the LBA algorithm

In Fig. 4, we use a generic fairness topology as was proposed in [24] to investigate the fairness of the algorithm when handling with connections with the same requirements and parameters but with different round trip times and traversed paths. All the links have a link rate of 1 Mb/s and are 1000 km long. The connection from source 1 to destination 1 (the transit source) traverses three links, all other connections traverse only one link (local connections). All sources have an initial sending rate of 700 kb/s and a bandwidth range of a maximum 1.5 Mb/s to 100 kb/s.

As all links are shared only between two connections we would expect every connection to get half the link capacity, i.e., 500 kb/s. However, Fig. 5(a) reveals that the rate of the transit connection actually gets reduced down to the minimum transmission rate of 100 kb/s, while the local connections get the remaining 900 kb/s.



(a) Bandwidth distribution



(b) Loss

Figure 5: Bandwidth and loss values measured at the receivers with adaptation applied

It is obvious that the LBA algorithm is unfair towards the long distance connection. This is due to a problem already noticed in TCP [9] and early ABR proposals [1], namely the beat down problem. The beat down problem arises as a result of the increase in the loss probability with each additional traversed link, see also [20]. Hence, the long distance sender receives control messages with higher loss values than the loss values reported to the local senders. Thereby, the transit connections tend to decrease their sending rates more often and with higher values than the local ones. Until now, no solution has been proposed for solving this problem in TCP. In the case of ABR, several solutions requiring the network switches to calculate the fair share and sending this value to the sources were introduced [12].

5 TCP and Adaptive Application Control

The transport control protocol (TCP) was designed primarily for loss sensitive applications without stringent delay requirements. On the contrary, applications using adaptive QoS control are less sensitive to data losses and have more stringent delay requirements. In this section, we investigate the effects of these differences and the differences in the way TCP and the LBA algorithm react to congestion in the network when TCP connections share a bottlenecked link with connections using an adaptive scheme. As a simulation topology we used the topology depicted in Fig. 6. The TCP source was based on 4.3BSD Reno TCP [23].

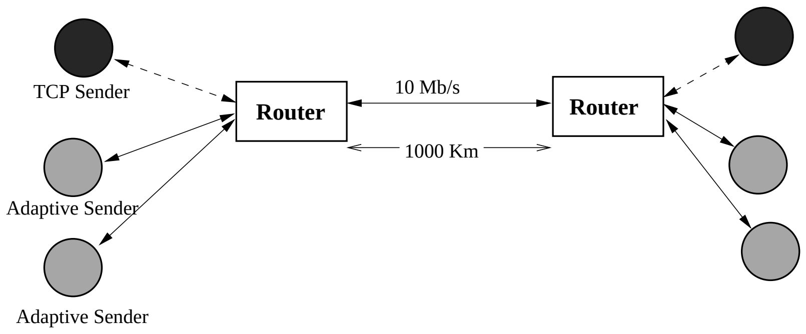


Figure 6: Test configuration for the interaction of the LBA algorithm and TCP

The results obtained from our simulations, see Fig. 7, suggest that the TCP connection, actually, gets starved and receives on the average only about 10% of the available bandwidth and that around 90% of the bandwidth is being taken by the adaptive connections. This, however, was to be anticipated. With the acknowledgment scheme of TCP, senders are usually informed about packet losses within a time period much smaller than the minimal time between two RTCP packets, i.e., 5 seconds. With each loss notification, TCP reduces its transmission window by half and for older TCP versions, e.g., Tahoe TCP even down to one packet. So, while the adaptive sources keeps on sending with the high transmission rate until an RTCP packet with loss indication arrives the TCP source reduces its transmission window and thereby its rate. That is, the adaptive sources can for a longer time period send data with the rate that might have actually caused the congestion. Also, as TCP reduces its transmission rate the congestion level will be decreased, so that the adaptive sources will finally have to react to a reduced congestion level. Finally, the adaptive scheme will only start reacting to congestion if the losses are larger than the loss threshold. TCP, on the contrary, reacts to any lost packets. Therefore, in Fig. 7 we can notice that during the first 200 seconds the TCP connection reduces its rate whereas the adaptive connections actually increases their transmission rate. The adaptive connections only start reacting to congestion after measuring a loss ratio higher than the loss threshold that was set here to 5%.

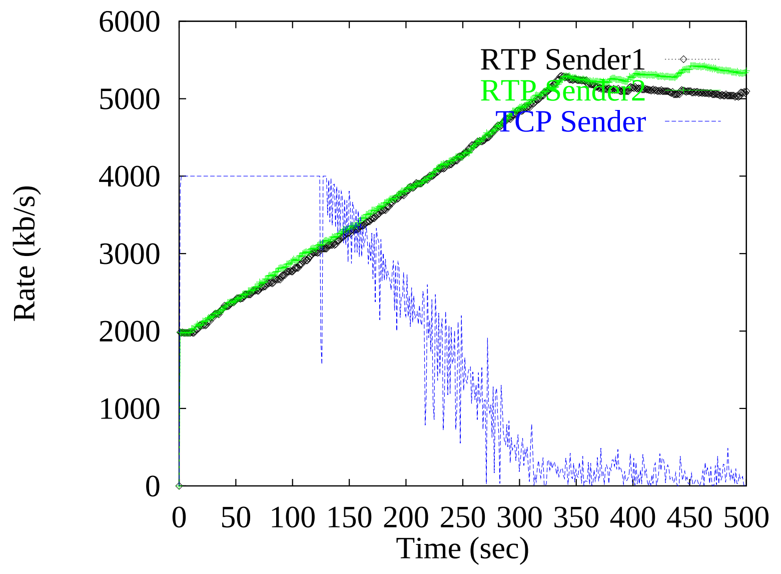


Figure 7: Bandwidth distribution between a TCP and connections using the LBA scheme

Different solutions have already been proposed to overcome this problem:

- Enhance UDP or UDP-based applications with TCP-like adaptation schemes.

That is, acknowledge each received data packet and use TCP's congestion control mechanisms to adapt the senders transmission rate to the network congestion state, see [11]. While this would be a fair approach towards TCP connections sharing the same bottlenecked connections with the adaptive connections, its usability is at best limited. In multicast sessions with large groups this would lead to acknowledgments implosion at the senders, as they receive an acknowledgment for each sent packet and each receiver. Also, this might be unnecessary as multimedia streams which would use such a scheme are not that loss sensitive and might not want to react to every single packet loss with a large rate reduction. In addition to that, rapidly changing the quality of a video stream to adapt to the network conditions might result in an annoying overall video quality. Finally, as TCP is unfair in the context described in Sec. 4, see [21, 9], that is in the terms of the max-min fairness criterion, this solution would only solve a part of the adaptive QoS control problems.

- Use the TCP-throughput model described in [10, 15] to adjust the transmission rate of the end systems. A proposal for an adaptation scheme based on this model, see [19], shows that such an approach results in a very oscillatory adaptation behavior and that the model needs further enhancements in order to use efficiently.

- Another possible solution is to map UDP and TCP connections into different traffic classes that are buffered in different queues at the routers. Such an approach is being discussed for the controlled load [25] service of the integrated service model [6].

- Finally, a solution that would ensure fairness for both TCP connections as well as adaptive sources without having to separate between streams, would be to choose an approach similar to that used in ABR. That is, introduce a new service class to the integrated service model that not only guarantees a minimal QoS level but dynamically informs the senders of the actual resources they could use. So, with such an approach the intermediate routers could calculate the fair bandwidth share each connection could use and inform the end systems -possibly using RSVP or RTCP- about their fair shares.

6 Summary and Future Work

In this paper, we presented a new approach for dynamically adjusting the sending rate of applications to the congestion level observed in the network. The senders can increase their sending rate during underload situations and then reduce it during overload periods. Results obtained through measurements suggest the efficiency of the algorithm in utilizing any free bandwidth in the network while maintaining a low loss level.

However, the results presented here reveal that the LBA scheme is unfair towards connections with longer round trip times or towards competing TCP connections. Our belief is that these problems are inherent to any adaptation scheme based on control information collected only at the end systems. Due to the beat down problem, the loss ratio observed for a given connection will usually increase with the number of traversed links. So, with the higher loss ratio the sources will tend to decrease their sending rates more often and thereby introduce the here observed unfairness. Work done on other adaptation schemes such as [4] show the same unfair behavior noticed in this work [8]. A possible solution might be to introduce a service class for adaptive applications similar to the ABR service of ATM [17]. That is, let the network nodes determine the exact fair bandwidth share a connection should use to avoid congestion and notify the sender of this fair share.

7 Acknowledgments

The RTP/RTCP simulation models were mainly implemented by Timur Friedman and improved by Frank Emanuel. The measurements were done by Krzysztof Samp. The comments of Henning Schulzrinne and Adam Wolisz are gratefully acknowledged and were the basis for various aspects of this work.

References

[1] A. Arulambalam, X. Chen, and N. Ansari. Allocating fair rates for available bit rate service in atm networks. *IEEE Communications*, 34(11):92 – 100, Nov. 1996.

[2] S. Baker. Multicasting for sound and video. *Unix Review*, 12(2):23–29, Feb. 1994.

[3] R. Braden, L. Zhang, and S. Berson. Resource reservation protocol (RSVP) – version 1 functional specification. Internet Draft, Internet Engineering Task Force, Nov. 1995. Work in progress.

[4] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP. *Computer Communications*, 19(1):49–58, Jan. 1996.

[5] A. Campbell, D. Hutchinson, and C. Aurrecoechea. Dynamic QoS management for scalable video flows. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science, pages 107–118, Durham, New Hampshire, Apr. 1995. Springer.

[6] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: architecture and mechanism. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 14–26, Baltimore, Maryland, Aug. 1992. ACM. *Computer Communication Review*, Volume 22, Number 4.

[7] C. Diot, C. Huitema, and T. Turletti. Multimedia application should be adaptive. In *HPCS*, Mystic, Connecticut, aug 1995.

[8] F. Emanuel. Investigation of scalability and fairness of adaptive multimedia applications. Studienarbeit, School of Engineering, TU Berlin, Berlin, May 1997. Work in progress.

[9] S. Floyd. Connections with multiple congested gateways in packet-switched networks, part 1–one-way traffic. *ACM Computer Communication Review*, 21(5), Oct. 1991.

[10] S. Floyd and F. Kevin. Router mechanisms to support end-to-end congestion control. Technical report, Feb. 1997.

[11] S. Jacobs and A. Eleftheriadis. Providing video services over networks without quality of service guarantees. In *RTMW'96*, Sophia Antipolis, France, Oct. 1996.

[12] R. Jain. Congestion control and traffic management in ATM networks: Recent advances and a survey. *Computer Networks and ISDN Systems*, Feb. 1995.

[13] I. Kouvelas, V. Hardman, and A. Watson. Lip synchronization for use over the internet: Analysis and implementation. In *GLOBECOM'96*, London, UK, Nov. 1996.

[14] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proc. of ACM Multimedia '95*, Nov. 1995.

[15] T. Ott, J. Kemperman, and M. Mathis. Window size behavior in tcp/ip with constant loss probability. In *The Fourth IEEE Workshop on the Architecture and Implementation of Hi gh Performance Communication Systems (HPCS'97)*, Chalkidiki, Greece, June 1997.

[16] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, Jan. 1996.

[17] S. S. Shirish. ATM Forum traffic management specification version 4.0. Technical Report 94-0013R6, ATM Forum, June 1995.

[18] D. Sisalem. End-to-end quality of service control using adaptive applications. In *IFIP Fifth International Workshop on Quality of Service (IWQOS '97)*, New York, May 1997.

[19] D. Sisalem and F. Emanuel. The direct adjustment algorithm: A tcp-friendly adaptation scheme. Technical report, Aug. 1997.

[20] D. Sisalem and H. Schulzrinne. End-to-end rate control in ABR. In *First Workshop on ATM Traffic Management (WATM'95)*, pages 281–287, Paris, France, Dec. 1995. IFIP WG 6.2.

[21] D. Sisalem and H. Schulzrinne. Congestion control in TCP: Performance of binary congestion notification enhanced TCP compared to Reno and Tahoe TCP. In *International Conference on Network Protocols (ICNP)*, pages 268–275, Columbus, Ohio, Oct. 1996.

[22] D. Sisalem, H. Schulzrinne, and C. Sieckmeyer. The network video terminal. In *HPDC Focus Workshop on Multimedia and Collaborative Environments*

(Fifth IEEE International Symposium on High Performance Distributed Computing), Syracuse, New York, Aug. 1996. IEEE Computer Society.

[23] W. R. Stevens. *TCP/IP illustrated: the protocols*, volume 1. Addison-Wesley, Reading, Massachusetts, 1994.

[24] L. Wojnaroski. Baseline text for traffic management sub-working group. Technical Report 94-0394r5, ATM Forum, Oct. 1994.

[25] J. Wroclawski. Specification of the controlled-load network element service, Nov. 1996. Internet Draft work in progress.