The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme*

Dorgham Sisalem
GMD-Fokus, Berlin
sisalem@fokus.gmd.de

Henning Schulzrinne
Columbia University, New York
schulzrinne@cs.columbia.edu

Abstract

Many distributed multimedia applications have the ability to adapt to fluctuations in the network conditions. By adjusting 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. In this paper, we present a new scheme called the loss-delay based adjustment algorithm (LDA) for adapting the transmission rate of multimedia applications to the congestion level of the network. The LDA algorithm was designed to reduce losses and improve utilization in a TCP-friendly way that avoids starving competing TCP connections. It relies on the end-to-end Real Time Transport Protocol (RTP) for feedback information. In addition, we enhanced RTP with functionalities for determining the bottleneck bandwidth of a connection. The bottleneck bandwidth is then used for dynamically determining the adaptation parameters. Simulations and measurements of the LDA algorithm suggest the efficiency of the scheme in utilizing network resources, decreasing loss ratios and its fairness towards competing TCP traffic. Our investigations also suggest that the report intervals and feedback information of RTP require some enhancements to accommodate the needs of adaptive QoS control schemes.

1 Introduction

A large part of the multimedia conferencing applications used currently in the Internet, such as VIC [17] and VAT [14] are based on the UDP transport protocol. However, UDP offers no quality of service control mechanisms and can therefore not guarantee any level of QoS. Fluctuations of network conditions combined with the inability of those protocols to support QoS control often render multimedia applications useless.

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

In addition, deploying non-congestion-controlled UDP traffic results in extreme unfairness towards competing TCP traffic. Sending best-effort traffic without any consideration of the network congestion state can easily result in packet losses. In response to that, TCP connections sharing the same bottleneck would reduce their transmission rates. However, without any rate reduction on behalf of the non-congestion-controlled traffic, the TCP connections which constitute around 95% of the Internet traffic today [25] would starve and receive much less than their fair bandwidth shares. Therefore, UDP sources need to be enhanced with quality of service control mechanisms that not only aim at reducing loss ratios and improve bandwidth utilization but also are fair towards competing TCP connections, i.e, be *TCP-friendly*.

Currently, different approaches are being discussed for solving the QoS and congestion control problems such as: resource reservation [4], priority mechanisms [1] and application control [7], i.e., to instruct the applications at the end systems to adapt the bandwidth share they are utilizing to the network congestion state.

Reserving enough resources for supporting a certain QoS level in advance guarantees this quality level and would be the most straightforward approach for handling the problem. However, as it is usually impossible to know the exact characteristics of a stream in advance, one would tend to over-allocate resources to guarantee the requested QoS level, leading to network underutilization. In addition to the complexity and scalability problems of reservation based QoS control schemes, these schemes do not easily allow the use of extra bandwidth for improved quality during network underload states for example.

With priority mechanisms, different data packets or streams are labeled with different priorities and thereby treated differently at the network routers. Such an approach is simpler than the reservation approach as it requires no signaling and less complicated control mechanisms at the routers. However, the exact mechanisms for setting the priority levels, the router mechanisms for controlling these levels and the actual gain achieved with such an approach are still under discussion [1].

In this paper, we propose to use QoS control mechanisms based on adapting the sender transmission rate in accordance with the network congestion state. That is, based on feedback information from the receivers, the sender would increase its transmission rate during underload situations and reduce it otherwise. This is especially beneficial when the bandwidth availability may change during a session, particularly during long-lived sessions typical for multimedia communications. Note, that with such an approach no QoS guarantees can be made, but the user perceived quality is improved through loss reduction.

Ergonomic studies and the experience gained from the Internet demonstrate that people can use audio and video data as long as the information content is above a minimum level [28]. This level depends on media content and the task at hand. For instance, a foreign language is more difficult to understand than a native language when audio quality is reduced. So, at the expense of slight degradation in user satisfaction, it is possible to adjust the transmission rates of end systems in accordance with the network congestion state. Hence, deploying application control results in a better overall bandwidth utilization, as it avoids network congestion. A major advantage of application control schemes is that they require no support from intermediate routers and are, thus, easy to deploy.

Applications control works best for live transmission for unicast and small multicast groups; in large multicast groups in a heterogeneous environment, a "race to the bottom" can occur so that one poorly connected receiver determines the quality for the much larger number of well-connected receivers. To avoid these problems, various proposals have been made for hierarchical data distribution [27, 18, 29]. Those proposals are based on partitioning a data stream into a base layer, comprising the information needed to achieve the lowest quality representation and a number of enhancement layers. The different layers are then sent to different multicast sessions and the receivers determine how many sessions to join and thereby adjust their QoS in respect to their own requirements and capacities.

While layered data transmission solves the heterogeneity problems, it might cause additional delays at the receivers. As the different layers might use different paths and hence have different round trip times, the receivers need to resynchronize the arriving data. Additionally, data partitioning might lead to a drift problem caused by motion compensated coding if only the lower bit rate layer is received [11]. Finally, it increases the complexity of the end systems considerably.

In this paper, we propose a new end-to-end rate adaptation scheme called the loss-delay based adjustment algorithm (LDA) for adjusting the transmission rate of multimedia applications to the congestion level of the network. The LDA algorithm resembles other adaptation approaches proposed in the literature [3, 5] that increase the transmission rate during network underload periods by an additive increase rate (AIR) and reduce the rate multiplicatively during congestion periods. However, these schemes show a rather aggressive adaptation behavior, often leading to the starvation of competing TCP traffic [23]. The LDA algorithm is on the contrary designed in a TCP similar fashion that prevents the starvation of TCP connections but sill allows for a stable transmission behavior.

Another issue that none of the adaptation schemes we encountered addresses is how to choose AIR. The appropriate value of AIR depends on the bandwidth share of a connection. As this share might change dynamically during the lifetime of a connection using a constant AIR value is rather inappropriate. The LDA algorithm adjusts its adaptation parameters dynamically in response to the losses, delays and capacity observed on the path traversed by the adaptive connection. The capacity of the network is estimated by enhancing RTP with the packet pair approach presented in [13].

In Sec. 2 of this paper we present some related work on the issue of TCP-friendly adaptation. The loss-delay based adjustment algorithm (LDA) is then presented in Sec. 3. Finally, the performance of the algorithm in terms of bandwidth utilization, scalability and fairness towards competing TCP connection is investigated using simulations and measurements in Sec. 4.

2 TCP-Friendly Adaptation Algorithms

As deploying non-congestion controlled UDP traffic in the Internet might result in starving competing TCP connections [10], different approaches have been suggested that aim at adjusting the transmission behavior of UDP senders in a TCP similar fashion. In the following we present a brief summary of some of those approaches:

- Jacobs [12] presents a scheme that uses the congestion control mechanisms of TCP, however, without retransmitting lost packets. In his scheme, the sender maintains a transmission window that is advanced based on the acknowledgments of the receiver. Based on the size of the window the sender can estimate the appropriate transmission rate. Thereby, the adaptive connection is guaranteed to acquire the same bandwidth a TCP connection would be getting under the same conditions of losses and delays. However, the need to acknowledge each packet limits the benefits of such an approach to unicast communication. Also, the acknowledgment packets increase the overall network traffic significantly diminishing the benefit of adaptation, particularly for short voice packets.

- Floyd et al. [10] and Ott et al. [19] propose a model that estimates the throughput of a TCP connection (r_{TCP}) under known delay and loss conditions as

$$r_{\text{TCP}} = \frac{1.22 \times M}{\text{RTT} \times \sqrt{l}} \tag{1}$$

with M as the maximum packet length, RTT as the round trip delay of the connection and l as the average loss measured during the lifetime of the connection.

Based on this estimation, Mahdavi et al. [15] and Turletti et al. [26] propose end-to-end, TCP-friendly congestion control schemes in which the end systems measure losses and delays in the network and restrict their transmission rates to the value estimated by Eq. 1. Floyd et al. [10] describe a mechanism in which routers identify connections that use more bandwidth than allowed by Eq. 1 and then throttle these connections.

The TCP-throughput model is based on a TCP sender interpreting packet drops as an indication of congestion and responding by reducing the congestion window and thereby the effective sending rate by half. Further, this sender should only increase its transmission window by at most one packet each round trip time.

While this model is rather simple, it is only partially correct. It does not consider timeout cases or delayed acknowledgments. However, even under those restrictions different studies [10, 16] have shown this model to be accurate enough for losses of less than 16%.

Note, however, that the TCP-throughput model is based on the *average* loss and delay observed during the lifetime of a connection. However, adaptation decisions need to be taken based on the current loss and delay values. As the observed losses and round trip delays change dynamically, using the TCP-throughput model as the sole basis for adaptation results in a rather oscillatory adaptation behavior which might lead to an annoying perceived QoS by the users due to the rapid changes in the quality of the received data stream. In addition, the TCP-model does not state the rate increase method during no loss periods. Testing a simple adaptation mechanism based on setting the bandwidth to the rate estimated by the TCP model using the loss and delay values reported in the RTCP packets and increasing the rate additively during no loss periods the scheme showed a very oscillatory behavior and on the average a rather low throughput [23]. Also, the authors in [19] state that if the round trip time is caused by queuing delays, or if the bottleneck router is shared among competing connections, this model is only of limited value.

3 The Loss-Delay Based Adjustment Algorithm (LDA)

The LDA algorithm is a sender based adaptation scheme. It relies on the Real Time Transport Protocol (RTP) [20] for feedback information about the losses at the receivers and round trip time. We additionally enhanced RTP to estimate the bottleneck bandwidth of a connection. Based on these information, the sender calculates for each received feedback message from receiver (i) the appropriate bandwidth to use for the reporting receiver (r_i). During periods without losses the sender increases r_i by an additive increase rate (AIR_i) which is estimated using the loss, delay and bottleneck bandwidth values included in the feedback reports. The actual sending rate is then determined at equally spaced adaptation points as $\min(r_i)$.

We divide the algorithm description into three parts handling the feedback information, upon which the adaptation decisions are taken, the mechanisms for setting the adaptation parameters and the actual adaptation mechanisms.

3.1 Feedback Information

The loss-delay based adjustment algorithm is based upon the Real Time Transport Protocol (RTP) [20] designed within the Internet Engineering Task Force (IETF). RTP enhances other transport protocols such as UDP with features needed by continuous media applications, such as the capability of distinguishing between different media streams and keeping track of various statistics describing the quality of the transmission 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 (using the sub-protocol called RTCP). Each session member periodically sends RTCP control reports to all other session members. Senders transmit reports describing the amount of data they have sent and a timestamp indicating the time the report was generated. For each incoming stream the receivers send a report indicating the fraction of lost data, the timestamp of the last received sender report (t_{LSR}) for that stream and the time elapsed in between receiving the last sender report and sending the receiver report (t_{DLSR}). Knowing the arrival time (t) of the RTCP packet the sender can calculate the round trip time (RTT) as follows:

$$\text{RTT} = t - t_{\text{DLSR}} - t_{\text{LSR}} \tag{2}$$

This calculation requires no synchronization between the clocks of the sender and receiver and is therefore rather accurate. With the LDA algorithm, we estimate the round trip propagation delay (τ) as the smallest measured RTT.

In addition, we enhanced RTP with the ability to estimate the bottleneck bandwidth of a connection based on the packet pair approach described by Bolot [2]. The essential idea behind this approach is: If two packets can be caused to travel together such that they are queued as a pair at the bottleneck, with no packets intervening between them, then the inter-packet spacing will be proportional to the time required for the bottleneck router to process the second packet of the pair.

We added to the RTCP packets an application defined part including the source sequence number, the sequence number (SEQ) of a data packet that will start a stream of probe packets and the number (n) of probe packets that will be sent. Then, n data packets starting with packet numbered SEQ are sent at the access speed of the end system. At the receiver site, the bottleneck bandwidth (b) is calculated as:

$$b = \frac{\text{probe packet size}}{\text{gap between 2 probe packets}}$$

Note, that we are using data packets as probe packets and therefore the additional bandwidth required for the bottleneck probing is restricted to the increased size of the RTCP packets. Sending data packets as probe packets is appropriate when transmitting video streams as one can send a few packets belonging to the same video frame together without considerably altering the traffic characteristics. Also video packets tend to be large which increases their possibility of being buffered at the routers. For the case of audio streams with small packets and stringent timing requirements another solution might need to be considered.

To estimate the average bottleneck bandwidth we need to further filter out incorrect estimates. We rely on an approach similar to that used in the BPROBE tool [6], namely clustering similar estimates into intervals, and choosing the average of the interval with the highest number of estimates. The estimated value is then sent back to the sender with the next receiver report.

Obviously, this approach has lots of drawbacks. We do not include the time between the transmission of two probe packets. But, as we send the packets at the sender's network access speed which in our case was 10 Mb/s we can usually ignore this time. Also, we do not consider packet drops or competing traffic. However, testing this approach on different Internet connections we achieved results comparable to those estimated by a more complicated tool such as the PATHCHAR[1] tool by LBL. Still, we need to further refine the estimates filtering method and do more measurements.

3.2 Dynamical Determination of the Additive Increase Rate (AIR)

The appropriate value to use for AIR depends largely on the bandwidth share a connection could utilize. For example, consider the case of a 1 Mb/s link shared between 2 connections. An ideal sharing would be if each connection received 500 kb/s. For this case, measurements [5] have shown that an appropriate AIR value that allows small convergence periods and only small oscillations would be 50 kb/s. This value would, however, be inappropriate if the link was being shared among 100 connections for example. With the LDA algorithm the additive increase rate (AIR) is set dynamically based on the congestion state of the network. That is, AIR is set initially to a small value, and is then increased during periods without losses. If losses are reported by the receivers, the senders set the AIR back to the initial value. We have chosen an initial value of 10 kb/s, which is small enough to be used even in narrowband ISDN connections.

If the received RTCP messages from receiver (i) indicate no losses the sender calculates an AIR_i for this receiver as follows:

$$\begin{aligned} \text{AIR}_i &= \text{AIR} + \text{AIR} * B_f \\ \text{with } B_f &= 1 - \frac{r}{b} \end{aligned}$$

with AIR as the additive increase value calculated for the entire session, r the current transmission rate, b the bottleneck bandwidth and B_f as a bandwidth factor that allows connections with a low bandwidth share to use larger AIR values and thereby converge faster to their fair bandwidth share. The difference between AIR and AIR_i will be explained in Sec. 3.3.

Further, AIR_i is limited maximally to the average rate increase (r_{incr}) a TCP connection would utilize during periods without losses in a time interval equivalent to the interval between two adaptation points (T). So for a round trip propagation delay of τ and a packet size of M, a TCP connection would increase its transmission window each τ by one packet size. Translated into transmission rate increase, r_{incr} would be

$$r_{incr} = \frac{M \times (\frac{T}{\tau} + 1)}{2}.$$

3.3 Congestion Control Mechanism

With each received RTCP packet from a member i the sender calculates a transmission rate (r_i) it would use if member i was the only member of the multicast session. r_i is then saved into a data base. So, if no losses were indicated the sender can recalculate AIR_i as was described in 3.2 and r_i is then set to

$$r_i = r + \text{AIR}_i$$

with r as the transmission rate the sender is using for the entire session.

[1] A tool for estimating the loss, delay and bandwidth characteristics of an Internet path. The tool is freely available from ftp://ftp.ee.lbl.gov/pathchar/

Otherwise, the sender reduces r_i in a rather TCP similar fashion, i.e., proportional to the indicated losses (l)

$$r_i = r * (1 - (l * R_f))$$

with R_f as the reduction factor. This factor determines the degree of the reaction of the sender to losses. Higher values result in a faster reduction of the transmission rate but a more oscillatory behavior. Lower values, on the other hand, lead to more stable rate values but result in longer convergence periods. The different simulations we ran suggest that we get the best tradeoff between convergence time and stability for a reduction factor between 2 and 5. Throughout this study we use a factor of 3.

Additionally, to favor connections with a low bandwidth share, we define a lower loss threshold (l_{th}) below which a connection with lower transmission rate than an *equivalent TCP connection* can increase its transmission rate. Equivalent TCP connection indicates here a TCP connection with the same packet size, round trip delay as can be calculated using the timing information of the RTCP packets and having the same losses. In all of our simulations we set l_{th} to 1%.

Instead of reacting to each RTCP packet as some adaptation schemes suggest [5], the LDA algorithm is based on so called adaptation points. At each adaptation point, the sender searches the transmission rates data base for the minimum value (r_{min}) and sets r to r_{min}. A r_{min} smaller than the current transmission rate (r) indicates that at least one member has reported losses. In this case, AIR is reinitialized to the initial value of 10 kb/s. If r_{min} was higher than the current transmission rate (r), then AIR is set to AIR$_i$ calculated for the member for which r_{min} was determined for.

We have used a period of 5 seconds between two adaptation points which is also the average value between generating two RTCP packets at the same source. As the time between two RTCP packets might actually be larger than 5 seconds, choosing this fixed value has the disadvantage that the adaptation decision might be taken based on the reports of only a subset of the session members and not all of them. However, the other alternative would be to wait for the reports from all members before adapting. While this might work for small groups, for larger groups the interval between two sent RTCP packets increases and thereby the adaptation decision will be taken less frequently and, thus, be less effective.

4 Performance of the Loss-Delay Based Adjustment Algorithm

When designing an adaptive control scheme, following goals need to be considered:

- the scheme should result in a low packet loss ratio,
- achieve high overall bandwidth utilization,
- fairly distribute bandwidth between competing connections,
- and scale for large multicast groups.

In this section, we investigate the performance of the LDA algorithm with regard to those different requirements using both simulations as well as measurements on a real network.

4.1 Fairness and Convergence of the LDA Algorithm

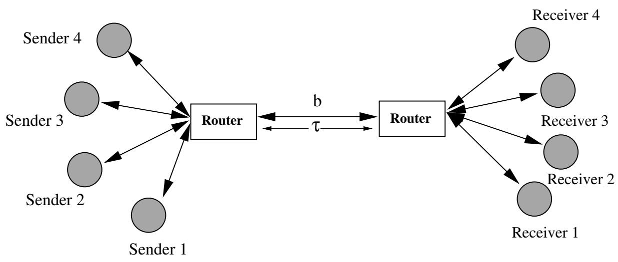
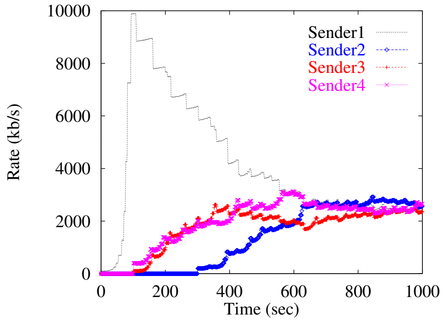
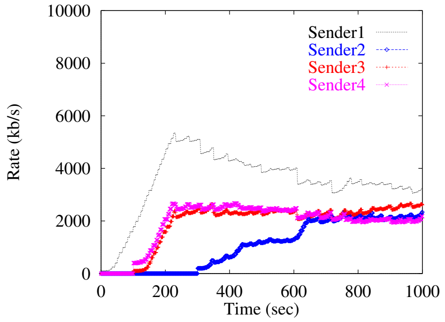


Figure 1. LDA performance testing topology

As a first performance test of the LDA algorithm, we simulated the topology depicted in Fig. 1. The model consists of 4 connections sharing a bottleneck router. All connections deploy the LDA algorithm and are persistent sources. That is, they always have data to send with the maximum allowed rate. They all have the same round trip propagation delay (τ) and are similar in their requirements and characteristics. The router is a random early drop (RED) gateway as was proposed by Floyd and Jacobson [9]. A RED gateway detects incipient congestion by computing the average queue size. When the average queue size exceeds a preset minimum threshold the router drops each incoming packet with some probability. Exceeding a second maximum threshold leads to dropping all arriving packets. This approach not only keeps the average queue length low but ensures fairness and avoids synchronization effects. Based on results achieved in [10] the minimum drop threshold was set to 0.5 of the routers buffer and the maximum one to 0.95 of the routers buffer. In our simulations we set the maximum queuing delay to 0.1 seconds and the bandwidth (b) of the bottleneck router to 1 and 10 Mb/s. The packet size is set to 1 kbyte, which is a typical video packet size. In the first set of simulations, we looked at the utilization, losses and fairness of the LDA algorithm with all connections having the same round trip time and different starting times. The bandwidth distribution and loss figures shown in Fig. 2 and 4 reveal that after a convergence period the four connections received equal bandwidth shares with average losses between

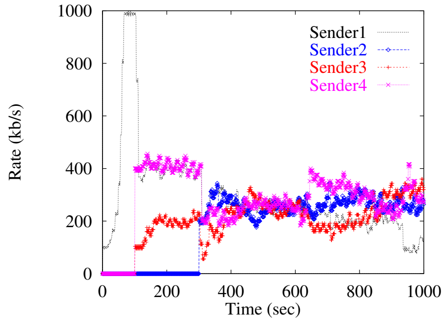


(a) τ = 1 ms, b=10 Mb/s

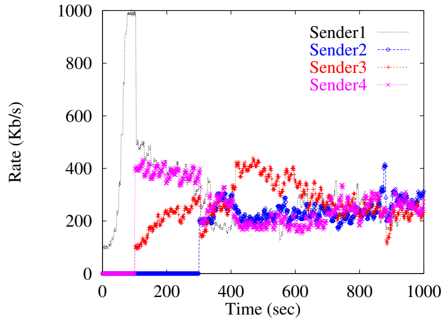


(b) τ = 100 ms, b=10 Mb/s

Figure 2. Rate of competing LDA connections with round trip propagation delay (τ) and link bandwidth (b)



(a) τ = 1 ms, b=1 Mb/s



(b) τ = 100 ms, b=1 Mb/s

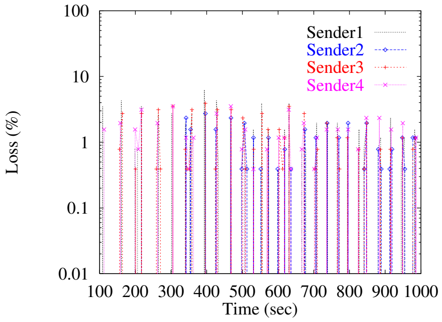
Figure 3. Rate of competing LDA connections with round trip propagation delay (τ) and link bandwidth (b)

2% and 10%. The utilization was around 95% in all measurements. Note that the convergence period for the simulations with bottleneck rate of 10 Mb/s, see Figs. 2(a) and 2(b), might extend over more than 300 seconds. This, however, depends on the value of the chosen reduction factor (R_f). We have used an R_f of 3. We could have achieved shorter convergence periods using higher values of R_f. This would, however, have lead to a more oscillatory adaptation behavior which is inappropriate for adaptive video as it would result in a lower perceived quality due to the rapidly changing video quality.

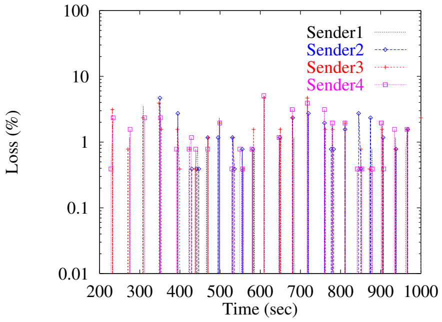
4.2 Interaction of TCP and LDA Traffic

The simulations run in Sec. 4.1 showed that the LDA algorithm achieves a rather fair bandwidth distribution among similar connections. In this section, we investigate the fairness of the LDA algorithm towards other adaptive traffic such as TCP. As a simulation topology, we use the topology depicted in Fig. 1 but with one of the senders replaced by a TCP sender. We investigate the bandwidth distribution achieved for the case of a bottleneck rate of 10 Mb/s and different round trip times.

The bandwidth distribution results depicted in Fig. 6 show that for the case of ($\tau = 1$ms) the TCP connection receives the same bandwidth share as the LDA connections. For ($\tau = 100$ms) the TCP connection receives nearly 1.4 Mb/s which is around half of its fair share which we would expect in this case to be 2.5 Mb/s. Based on Eq. 1 for a TCP connection with a packet size of 1 kbyte, a round trip time of 0.1 seconds to reach a bandwidth share of 2.5 Mb/s we would need to maintain an average loss rate of less than 0.16%. Actually, as we should also consider the queuing delay the loss value is even smaller. However, the RTCP packets only include a field of 8 bits for the loss value. That is, the minimum loss value that can be reported is around 0.4%. With such a loss value a TCP connection would utilize for the delay and packet size at hand a bandwidth share of around 1.6 Mb/s which is only slightly more than the value depicted in Fig. 6. To achieve optimal fair

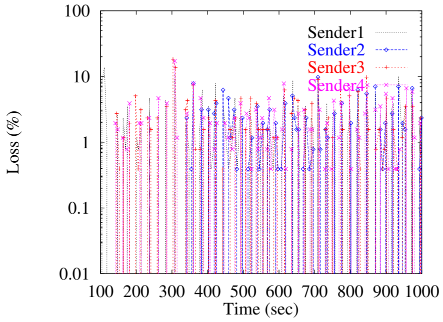


(a) τ = 1 ms, b=10 Mb/s

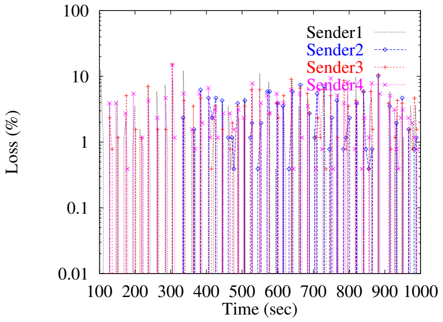


(b) τ = 100 ms, b=10 Mb/s

Figure 4. Loss of competing LDA connections with round trip propagation delay (τ) and link bandwidth (b)



(a) τ = 1 ms, b=1 Mb/s



(b) τ = 100 ms, b=1 Mb/s

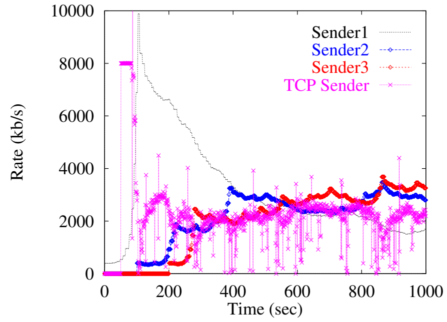
Figure 5. Loss of competing LDA connections with round trip propagation delay (τ) and link bandwidth (b)

bandwidth distribution with the TCP connection getting exactly the same share as the LDA connections we would need to enhance RTCP to carry finer granulated loss values. Another approach would be to use the value of the cumulative number of lost packets (l_{cum}) included in the RTCP packets instead of the loss fraction. This value indicates the number of packets lost since the beginning of reception. While this leads to more accurate loss indications it also increases the complexity of the scheme as the sender needs to maintain more state information to actually benefit from l_{cum}.

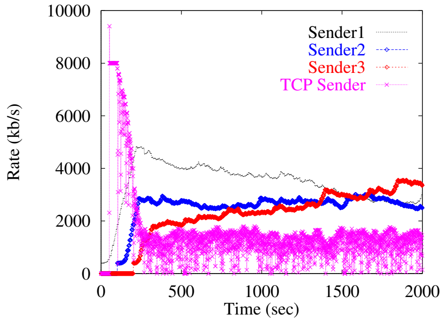
4.3 Scalability of the LDA Algorithm

To investigate the behavior of the LDA algorithm when used in multicast groups of different sizes we tested a simple case of a sender transmitting data to n receivers with n set to 1, 4, 64 and 320. As Fig. 7(a) shows the bottleneck router is shared between the sender and an on-off connection introducing bursty back ground traffic, see Fig. 7(b).

From Fig. 7(c) we can notice, that with a session of 64 or 320 members the reactions to the sudden loss peaks are larger than those for smaller groups. The reason for this is related to the RTCP scaling algorithm. To avoid the increase in control traffic with the increase in group size the RTCP traffic is restricted to 5% of the bandwidth utilized by the session. With 320 members in the session the interval between two successive RTCP messages is larger than the adaptation interval of 5 seconds that we are using. As the reported losses in an RTCP message is calculated over the entire time between the sending of two RTCP packets the effects of a loss peak will be noticed over several adaptation points even though the congestion has actually cleared. As an example, consider the case depicted in Fig. 8. Between the first two adaptation points we measure a loss rate of 10% over a time period of say 2 seconds. Assume, that the RTCP intervals of two members are 10 seconds long and are overlapping with one RTCP message arriving before the second adaptation point and the RTCP message of the other member arriving only before the third adaptation point. As the loss is averaged over the entire 10 seconds the first mem-

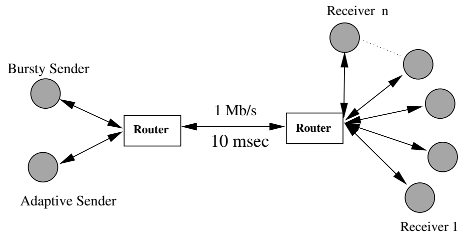


(a) τ = 1 ms, b=10 Mb/s

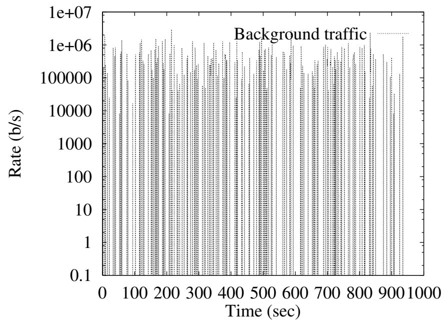


(b) τ = 100 ms, b=10 Mb/s

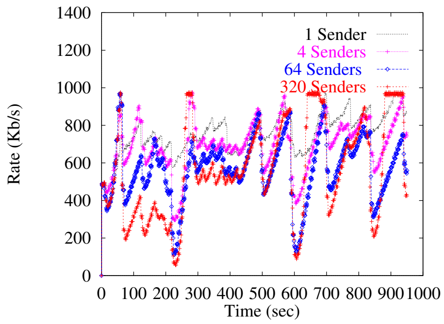
Figure 6. Bandwidth of LDA connections competing with a TCP connection with round trip propagation delay (τ) and link bandwidth (b)



(a) Multicast test topology for LDA



(b) Background traffic



(c) Sender transmission rate

Figure 7. Testing the scalability of the LDA algorithm

ber would report a loss of 2%. Thereupon, the sender would decrease its transmission rate and we would get a loss rate of 0% again. However, the second member will still report a loss of 2% even though no losses can be measured anymore. One approach that avoids this problem would be for the receivers to inform the senders about the losses seen in the last 5 seconds or so and not the average loss measured between the sending of two RTCP packets. This problem is being discussed in the IETF for a future version of RTP

4.4 Fairness of the LDA Algorithm

In this section, we investigate the fairness of the LDA algorithm for the case of:

- different connections sharing the same bottleneck but having different round trip times,
- and connections having the same round trip time but traverse a different number of congested links.

As TCP's throughput is inversely proportional to the round trip time we would expect TCP to be unfair under these conditions, that is two similar connections sharing the same bottleneck but having different round trip delays would not receive the same bandwidth share. We use the same topology depicted in Fig. 1 but with all senders starting to send data at the same time and the connections from senders 3 and 4 to the receivers 3 and 4 having a round trip time 10 times as large as the other two connections. The bandwidth distribution results depicted in Fig. 9 suggest that

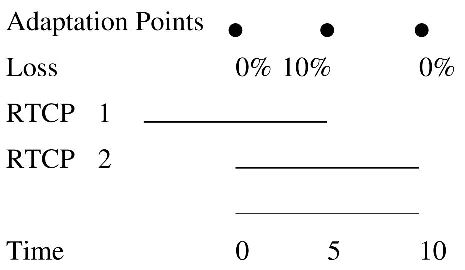


Figure 8. Timeline of adaptation points and RTCP report intervals

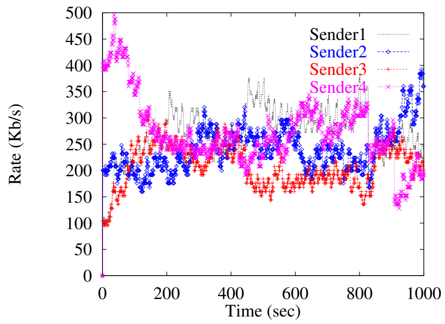
like TCP, the bandwidth distribution is unfair. This is especially evident for the case of a bottleneck bandwidth of 10 Mb/s. Due to the different round trip times the connections calculate different maximum additive increase rates. Additionally, for the case of losses below the lower threshold the transmission rate is set at the maximum to the TCP rate under comparable conditions. Thereby, connections with higher τ have a smaller maximum allowed rate and can thus only benefit less from this low loss situation.

Various studies [8, 22] have shown that the bandwidth of a TCP connection not only depends on the round trip delay but also on the number of traversed hops. This phenomena called the beat down problem arises as a result of the increase in the loss probability with each additional traversed link. In Fig. 10 we investigate the case of two connections having the same round trip delay with one connections traversing only one router and the the other one traversing two routers. Both connections use the LDA algorithm with the same initial parameters (l_{th}=1%, R_f=3 and initial AIR=10 kb/s). The results achieved for the router capacity of 1 Mb/s and 10 Mb/s, see Fig. 11, reveal that the connection traversing only one hop actually receives around twice as much bandwidth compared to the connection traversing two hops. Looking at the losses, see Fig. 12, the two connection suffer from we notice that the connection traversing two routers suffers higher losses and receives thereby a lower bandwidth share.

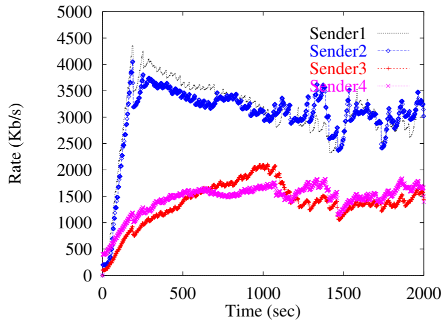
4.5 Measurement Based Testing

To verify the performance of the LDA algorithm we ran a simple experiment over our local network. We connected two workstations over a router with a limited capacity of 1 Mb/s. From the first station we sent two JPEG video streams using the NEVIT video tool [24] which we enhanced with the LDA algorithm. In addition, we started an FTP transfer from the first station to the second in parallel to the video streams.

The first 300 seconds of our measurements resemble to



(a) τ= 100, 1000 msec, b=1 Mb/s



(b) τ = 100, 1000 msec, b=10 Mb/s

Figure 9. Bandwidth distribution of LDA controlled connections with different round trip times propagation delays(τ) and link bandwidth (b)

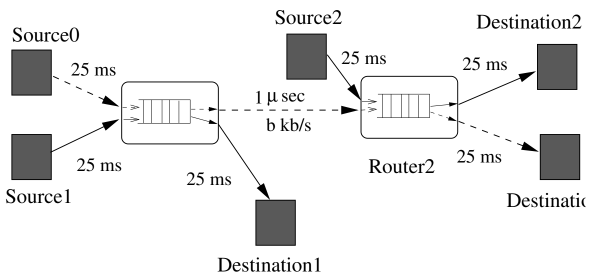
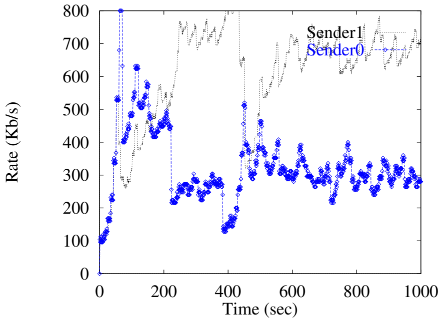
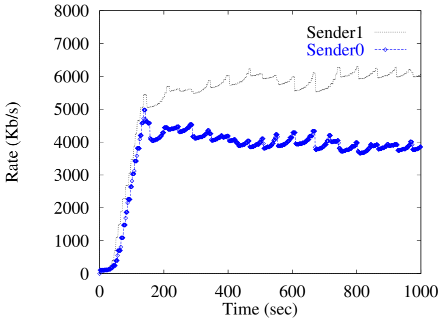


Figure 10. Test configuration for investigating the effects of the beat down problem on the LDA algorithm

some extent a typical situation we might encounter in the Internet. Both video streams were sent during this period without using the adaptation scheme. This results in the starvation of competing TCP connections and the UDP connec-

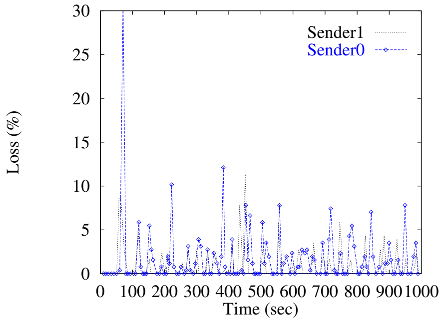


(a) τ= 100 msec, b=1 Mb/s

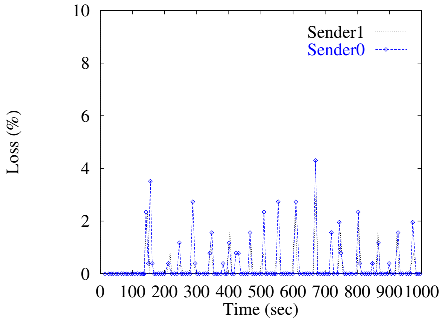


(b) τ = 100 msec, b=10 Mb/s

Figure 11. Bandwidth distribution of LDA controlled connections with different round trip propagation delays (τ) and link bandwidth (b)



(a) τ= 100 msec, b=1 Mb/s



(b) τ = 100 msec, b=10 Mb/s

Figure 12. Losses of LDA controlled connections with round trip propagation delay (τ) and link bandwidth (b) -Sender 0 traverses 2 hops and Sender 1 traverses 1

tions suffering high losses (around 50%). In the next 900 seconds both UDP connections adapt their transmission rate to the loss level. This results in a rather fair bandwidth distribution of around 300 kb/s for both the adaptive connections as well as the TCP connection. Additionally, the losses of the UDP connections drop to around 7%. In the final part of the measurement, one UDP connection stops sending data. We notice that both the other UDP connection and the TCP connection increase their bandwidth share up to around 500 kb/s.

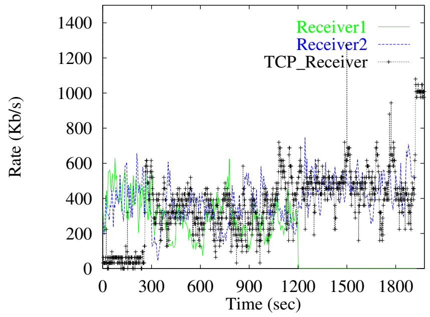
5 Summary and Future Work

In this paper, we presented a new approach (LDA) for dynamically adjusting the sending rate of applications to the congestion level observed in the network. The various simulations and measurements we made suggest the efficiency of the LDA algorithm in utilizing network resources and reducing losses. Also, the scheme is shown to be fair towards TCP traffic and flexible in its estimation of the appropriate adaptation parameters to use.

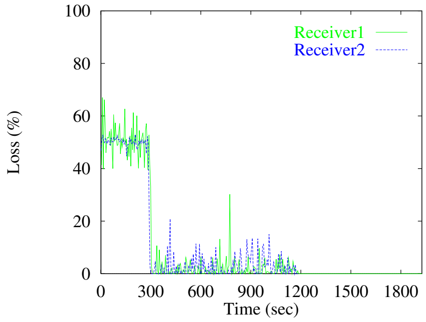
To address the problem of adaptation in heterogeneous environments, we are currently investigating a method for integrating sender based adaptation schemes such as the LDA algorithm with layered transmission methods. With such an approach, the sender determines the number of layers to use and the transmission rate to use for each layer dynamically based on the feedback information sent by the receivers [21].

Additionally, we are currently testing the efficiency of the bottleneck probing method we introduced in the LDA algorithm as well as the acceptability of adaptive video traffic from the users' point of view.

A major point that we did not address in this paper is how to enforce end systems to use adaptive QoS control schemes. In an environment in which most of the users behave in a rather social way and adapt their transmission rates to the congestion state of the network, employing non-adaptive transmission behavior would actually result in a better band-



(a) Bandwidth Distribution



(b) Losses

Figure 13. Measurement of bandwidth distribution and losses for the interaction of TCP and the LDA algorithm

width share for the offending sender.As a solution to this problem we could imagine a scenario in which the end systems deploy the LDA algorithm and the routers identify non-congestion controlled connections [10] and throttle them.

6 Acknowledgments

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

References

[1] S. Blake. Some issues and applications of packet marking for differentiated services. Internet Draft, Internet Engineering Task Force, Dec. 1997. Work in progress.

[2] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In D. P. Sidhu, editor, *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 289–298, San Francisco, California, Sept. 1993. ACM. also in *Computer Communication Review* 23 (4), Oct. 1992.

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

[4] 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.

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

[6] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical Report BU-CS-96006, Computer Science Departement, Boston University, Mar. 1996.

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

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

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

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

[11] U. Horn and B. Girod. Scalable video coding for the internet,. In *8th Joint European Networking Conference*, Edinburgh, England, May 1997.

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

[13] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, Aug. 1988. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.

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

[15] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control, June 1997. Technical note, available from http://ftp.ee.lbl.gov/floyd/papers.html.

[16] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *IEEE Network*, 11(6), November/December 1997.

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

[18] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *SIGCOMM Symposium on Communications Architectures and Protocols*, Palo Alto, California, Aug. 1996.

[19] 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.

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

[21] D. Sisalem and F. Emanuel. QoS control using adaptive layered data transmission. In *IEEE Multimedia Systems Conference'98*, Austin, TX, USA, June 1998.

[22] 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.

[23] D. Sisalem, H. Schulzrinne, and F. Emanuel. The direct adjustment algorithm: A TCP-friendly adaptation scheme. Technical report, GMD-FOKUS, Aug. 1997. Available from http://www.fokus.gmd.de/usr/sisalem.

[24] 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.

[25] K. Thompson, G. J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6):–, November/December 1997.

[26] T. Turletti, S. F. Prisis, and J.-C. Bolot. Experiments with a layered transmission scheme over the Internet. Rapport de recherche 3296, INRIA, Nov. 1997.

[27] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *INFOCOM'98*, San Francisco, USA, Mar. 1998.

[28] F. Wilson, I. Wakeman, and W. Smith. Quality of service parameters for commercial application of video telephony. In *Human Factors in Telecommunication Symposium*, Darmstadt, Germany, Mar. 1993.

[29] L. Wu, R. Sharma, and B. Smith. Thin streams: An architecture for multicating layered video. In *The 7th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 97)*, St. Louis, USA, May 1997.