MLDA: A TCP-friendly Congestion Control Scheme for Heterogeneous Multicast Environments

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

Adam Wolisz
Technical University Berlin
wolisz@ee.tu-berlin.de

Abstract

To avoid overloading the Internet and starving TCP connections, multimedia flows using non-congestion controlled UDP need to be enhanced with congestion control mechanisms. In this paper, we present a TCP-friendly adaptation algorithm called MLDA for adjusting the transmission rate of multimedia senders in accordance with the network congestion state. For taking the heterogeneity of the Internet and the end systems into account, MLDA senders support hierarchical data transmission where the shape and number of the layers is determined dynamically based on feedback information generated by the receivers. While MLDA is basically based on the real time transport protocol (RTP) we had to redesign RTP to allow for a more scalable and timely flow of feedback information from the receivers to the sender. Additionally, MLDA supports a scalable round trip delay estimation approach based on a combination of end-to-end measurements and one way measurements between possibly asynchronous end systems. The performance of MLDA is investigated using simulations and measurements.

1 Introduction

While congestion controlled TCP connections carrying time insensitive FTP or WWW traffic still constitute the major share of the Internet traffic today [23], recently proposed real-time multimedia services such as IP-telephony and group communication will be based on the UDP protocol. While UDP does not offer any reliability or congestion control mechanisms, it has the advantage of not adding delays to the carried data due to retransmissions as is the case with TCP. Additionally, as UDP does not require the receivers to send acknowledgments for received data UDP is well suited for multicast communication. However, deploying UDP in the Internet on a large scale might result in extreme unfairness towards competing TCP traffic. In response to losses in the network, TCP connections sharing the same congested links with UDP flows reduce their transmission rates. However, without any rate reduction on behalf of the non-congestion-controlled traffic, the TCP connections would starve and receive a much smaller bandwidth share than the competing UDP flows. Therefore, UDP flows need to be enhanced with control mechanisms that not only aim at avoiding network overload but are also fair towards competing TCP connections, i.e, be TCP-friendly. TCP-friendliness indicates here, that if a TCP connection and an adaptive flow with similar transmission behaviors have similar round trip delays and losses both connections should receive similar bandwidth shares. As an oscillative perceived QoS is rather annoying to the user, multimedia flows require stable bandwidth shares that do not change on the scale of a round trip time as is the case of TCP connections. It is, thus, expected that a TCP-friendly flow would acquire the same bandwidth share as a TCP connection only averaged over time intervals of several seconds or even only over the entire life time of the flow and not at every time point [5].

In this paper, we describe a general TCP-friendly congestion control approach for handling the case of heterogeneous multicast groups called multicast enhanced loss-delay based adaptation algorithm (MLDA). MLDA is based on the real time transport protocol (RTP) [17] widely used for multimedia communication in the Internet for carrying control information between the senders and receivers. With MLDA the receivers collect information about the losses, delays and bottleneck bandwidths on the paths connecting them to the sender and determine a bandwidth share that would be utilized be a competing TCP connection traversing the same path under the same loss and delay conditions. The sender periodically collects the determined bandwidth shares and adjusts its transmission rate in accordance with those values. For the case of heterogeneous receivers that vary in their processing capacities and Internet connections a single sender transmission rate cannot satisfy the conflicting bandwidth requirements at the different sites. This limitation can be overcome using layered transmission mechanisms as was proposed in [9, 25]. 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. As multicast routers only forward data requested by the receivers, the receivers can determine how many sessions to join and thereby adjust their QoS in respect to their own requirements and capacities.

Adaptive schemes using layered data transmission usually assume statically set layer sizes. However, to better accommodate the actual heterogeneity of the receivers, with MLDA the sender adjusts the sizes of the different layers it is transmitting based on the receiver feedback and informs the receivers about the sizes and addresses of the different layers. Knowing the sizes of the different layers and the TCP-friendly rate they could be using the receivers can then determine the number of layers to join.

To allow for scalable feedback and yet provide the sender with enough information about the actual heterogeneity of the receivers we introduce a novel feedback approach called "partial suppression" that allows RTP to exchange control messages in a more timely manner and suppress the transmission of reports carrying similar information. Additionally, to get an estimation of the round trip delay between the sender and receivers we propose a simple approach based on a combination of one way delay estimation using timestamps of non-synchronized hosts connected by possibly asymmetrical links and end-to-end measurement of the delay.

For determining the TCP-friendly bandwidth share at the receivers we use an algorithm we presented in a previous work called loss-delay based adaptation algorithm (LDA) [19].

In Sec. 2 we present a brief overview of some related work in the area of TCP-friendly congestion control for unicast and multicast environments. Sec. 3 describes MLDA and the enhancements needed for RTP as well as the rate estimation algorithm. In Sec. 4 and 5 we investigate through simulations and measurements the ability of MLDA to accommodate the needs of heterogeneous multicast receivers in a TCP-friendly manner.

2 Background and Related Work

Recently, there has been several proposals for TCP-friendly adaptation schemes that either use control mechanisms similar to those of TCP or base the adaptation behavior on an analytical model of TCP.

Rejaie et al. present in [15] an adaptation scheme called rate adaptation protocol (RAP). Just as with TCP, sent packets are acknowledged by the receivers with losses indicated either by gaps in the sequence numbers of the acknowledged packets or timeouts. Using the acknowledgment packets the sender can estimate the round trip delay. If no losses were detected, the sender can periodically increase its transmission rate additively as a function of the estimated round trip delay. After detecting a loss the rate is reduced by half in a similar manner to TCP.

Padhye et al. [12] present an analytical model for the average bandwidth share of a TCP connection (r_{TCP})

$$r_{\mathrm{TCP}} = \frac{M}{t_{\mathrm{RTT}}\sqrt{\frac{2Dl}{3}} + t_{\mathrm{out}} \min\left(1, 3\sqrt{\frac{3Dl}{8}}\right) l\left(1 + 32l^2\right)} \quad (1)$$

with M as the packet size, l as the loss fraction, t_{out} as the TCP retransmission timeout value, t_{RTT} as the round trip delay and D as the number of acknowledged TCP packets by each acknowledgment packet.

Using this model Padhye et al. [13] present a scheme in which the sender estimates the round trip delay and losses based on the receiver's acknowledgments. In case of losses, the sender restricts its transmission rate to the equivalent TCP rate calculated using Eqn. 1 otherwise the rate is doubled. While the scheme behaves in a TCP-friendly manner during loss phases, its increase behavior during underload situations might result in unequal bandwidth distribution due to the possibility of increasing the transmission rate faster than a competing TCP connection.

More appropriate for multicast communication, Handley et al. present in [6] a scheme in which the receivers estimate with Eqn. 1 the rate the sender should be using and inform the sender about this value. As this approach relies solely on the TCP-model, the receivers always need to estimate a loss value to be able to use Eqn. 1. This is achieved by making loss measurements over long time intervals.

Vicisano et al. [25] present a control scheme supporting layered streams. Here, the receivers join or leave a layer based on their measured loss value. Using specially flagged packets the sender indicates synchronization points at which receivers might join or leave a specific layer.

Bolot et al. [24] and Tan et al. [22] present schemes in which the sender transmits data in layers and the receivers calculate the rate appropriate for them using an analytical TCP model [5]. Based on this calculation they join the appropriate number of layers.

3 Multicast Enhanced Loss-Delay Adaptation Algorithm (MLDA)

Several aspects need to be considered when designing a congestion control mechanisms for multicast communication:

Scalability: The performance of the control scheme should not deteriorate with increasing numbers of receivers. Additionally, the amount of data gathered at the end systems or transmitted between the end systems should be sustainable within the available resources.

Heterogeneity: Internet links as well as the end systems connected to the Internet vary greatly in their capabilities and resources. Multicast congestion control schemes need to take this heterogeneity into account and aim at satisfying the requirements of a large part if not all possible receivers. So, for the case of n receivers one might actually determine a set of $[r_1, r_2, ..., r_n]$ rates the sender needs to adjust its transmission rate simultaneously to, to satisfy the needs of all receivers. This might be achieved by simulcasting the same content at the different rates [8] or referring to hierarchical data transmission by dividing a data stream into a base layer representing the transmitted data content in a basic quality and several enhancement layers that combined with the basis layer improve the overall perceived quality of the data stream. Each layer is then sent on a different multicast group. For the case of n receivers with $[r_1, r_2, ..., r_n]$ possible rates with r_i as an increasing value, the sender might set the rate of the lower layer to r_1, the first enhancement layer to $r_2 - r_1$ and so on. Each receiver would then join up to n layers based on its capacity.

The simplest approach for a congestion control scheme to support multicast communication would be for the sender to determine the appropriate transmission rate towards each receiver based on loss and delay information collected and sent by that receiver. At specific time points the sender would then set its transmission rate based on the rates determined for the different receivers. However, this approach has the following drawbacks:

- The amount of state information the sender needs to maintain increases linearly with the number of receivers.
- With such an approach 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.
- The work presented in [19] reveals that using the RTP control feedback approach, the performance of the adaptation scheme degrades with the increased number of receivers even for the case of homogeneous receivers, i.e, receivers with similar capacities connected over similar links to the sender. This is due to the loss multiplicity problem [2] with which the sender receives multiple loss indications for the same loss and reacts several times to the same congestion situation even though the first reaction was sufficient.

3.1 A General Description of MLDA

MLDA is basically a sender-based adaptation scheme. That is, the sender adjusts its transmission behavior in accordance with feedback information from the receivers. In contrast to other adaptation schemes such as [13, 15] that introduce a new protocol for establishing the flow of control messages between the receivers and the sender, MLDA is based on the control part of real time transport protocol (RTP) [17] called RTCP.

To avoid having to collect state information about all receivers at the sender, with MLDA we place the calculation of the transmission rate at the receivers. Hence, the receivers measure loss, delay and bottleneck bandwidth on the path connecting them to the sender and calculate a bandwidth share a competing TCP connection would be utilizing under the same conditions. The sender collects the calculated bandwidth shares and adjusts its transmission behavior based on those values. To better accommodate the needs of heterogeneous receivers we refer to using layered data transmission. That is, the sender divides its data stream into separate layers with the sizes of the layers determined using the feedback information from the receiver. The basic message exchange is as follows:

1. The sender transmits periodically reports containing the usual RTCP data and information about the sent layers.

2. After receiving a sender report each receiver (i) measures the loss and delay of the incoming data for a period of time and determines a TCP-friendly bandwidth share (r_i) the sender could utilize on the path connecting the sender and receiver.

3. Based on the calculated share and the rates of the layers as reported in the sender reports the receiver decides to join a higher layer, stay at or leave the current one.

4. Afterwards, the receivers schedule the transmission of reports indicating their calculated bandwidth share after a random period of T_{wait}. If a report form another receiver with rate indication similar to r_i was seen before T_{wait} expires the receiver suppresses the transmission of its report.

5. Based on the receiver reports, the sender adjusts the sizes of the different layers.

3.2 Sender Behavior

With RTCP the receivers usually set the time in between the sending of two reports in accordance with the number of members of the multicast session and the bandwidth dedicated for the control data. For large sessions this might result in very scarce reports not enough for achieving efficient adjustment of the sender behavior. With MLDA the sender polls feedback information from the receivers by periodically sending a sender report at intervals of $(T_{\mathrm{SR}} + p)$ with p as a uniformly distributed random variable that assures that the reports of senders that start at the same time do not get synchronized. In addition to the usual information such as a timestamp and amount of sent data, the sender includes in its reports information about the number of layers it is transmitting, the rate ($R_{\mathrm{L_k}}$) of each layer (L_{k}) and the address on which each layer can be received on.

In between the sending of two reports, the sender collects feedback information from the receivers. Based on these information, the sender adapts the sizes of the transmitted layers in accordance with the observed heterogeneity of the receivers before sending its next report.

3.2.1 Data Layering

Ideally, the sender would partition its data steam into a subset of layers that satisfy the needs of all receivers. In a communication scenario with n receivers reporting transmission rates of $(r_1, ..r_n : r_{\text{i-1}} < r_i < r_{\text{i+1}})$ the sender would then need to partition its data steam into subintervals $([r_1, r_2), [r_2, r_3), \cdots, [r_{\text{n-1}}, r_n)$. Each receiver determining r_x as its available bandwidth share would then join m layers with $(r_x = \sum_1^m r_i)$.

However, using a large number of layers might result into drift problems [7], increased delays due to the need to synchronize the different layers and higher complexity at the receivers. As an approximation usually only a few layers are used.

Here we refer to a simple approach for dividing the data into n layers. The sender determines the minimum (r_{min}) and maximum (r_{max}) reported rates and sets the rate for the basic layer to r_{min}. The enhancement layers are then set to $(\frac{(\alpha \times r_{\text{max}}) - r_{\text{min}}}{n-1})$. $(\alpha : \alpha \leq 1)$ is a dampening factor that reduces the effects of a possible overestimation of the available resources by the receiver reporting r_{max}. To avoid establishing layers with a negligible content, the minimum size of a layer is set to L_{min}.

Note, that any other approach for dividing the data in different layers could be used instead. The layers could be chosen based on the coding used or the number of receivers requesting a certain rate if such information are available [1].

3.3 Receiver Behavior

After receiving a sender report the receivers measure the losses of the incoming data stream for a period of $(T_{\text{o}} \times x)$ with T_{o} as the minimum measurement time needed to obtain usable loss information. x indicates the number of layers the receiver is already tuned to. After the measurement period each receiver (i) calculates the rate (r_i) that would be appropriate to use on the link connecting the sender to him. With R_{L_k} as the transmission rate on layer L_k the receiver can now take one of the following actions:

$r_i \geq \sum_0^{x+1} R_{\text{L}_\text{k}}$: The determined TCP-friendly rate is higher than the rate of the incoming data stream in addition to the rate of the next higher layer. The receiver joins the next higher layer and starts an observation time of T_{o} to measure the loss values of the now increased incoming stream. After the measurement period a new transmission rate is determined and the receiver can again join a higher layer, leave the just joined layer or stay at the higher layer.

$r_i < \sum_0^{x} R_{\text{L}_\text{k}}$: In this case the rate of the incoming data stream is higher than the theoretically TCP-friendly share determined with an adaptation algorithm. The receiver must, thus, leave the highest layers it is currently receiving until the inequality $(r_i \geq \sum_0^x R_{\text{L}_\text{k}})$ is satisfied.

$\sum_0^x R_{\text{L}_\text{k}} < r_i < R_{\text{L}_{\text{m+1}}}$: The receiver stays at the current level.

Finally, the receiver schedules the transmission of a report carrying the value of r_i in addition to the loss and delay information usually carried in receiver reports.

Note that, joining a multicast session can be thought of as increasing the bandwidth consumed by the receiver instantously through the addition of a new layer. However, leaving a session only results in a reduction in the rate as measured at a receiver after a period of time. The network routers that forward the multicast data to some receiver need to make sure that the session is no longer requested by any receiver before stopping sending the data on the links towards that receiver. This latency in the leave procedure needs to be considered by schemes such as [9, 25] that determine the appropriate number of layers to join by observing the losses for a period of time and leaving the highest layer is case losses were observed and then repeating the same procedure. As leaving a layer results only in a reduction of the consumed bandwidth after a period of time, the receiver should not restart measuring the losses until the leave procedure has taken effect. This significantly increases the time required for such schemes to adapt to changes in the network conditions.

With MLDA the algorithm determines directly the number of layers a receiver can be listening to. Hence, there is no need for observing the network situation after dropping a layer and taking the leave latency into account.

3.4 Enhancements to the Real Time Transport Protocol

RTP [17] defines a data and a control part. For the data part RTP specifies an additional header to be added to the data stream to identify the sender and type of data. With the control part (RTCP), each member of a multicast session periodically sends control reports to all other members containing information about sent and received data. Additionally, the end systems might include in their reports an application specific part (APP) intended for experimental use. The RTCP traffic is scaled with the data traffic so that it makes up a certain percentage of the data rate (usually 5%) with a minimum interval of 5 seconds between sending two RTCP messages. RTCP messages include information about the losses and delays noticed in the network.

To better accommodate the varying needs of heterogeneous receivers and provide more information about the available resources in the network we enhanced the feedback mechanisms of RTCP and extended RTP with the ability to estimate the bottleneck bandwidth of a flow.

3.4.1 Bottleneck Bandwidth Measurement with RTP

To estimate the maximum possible bandwidth share a flow can utilize we enhanced RTP with the ability to estimate the bottleneck bandwidth, i.e., the capacity of the smallest router, on the path connecting the sender to the receiver. The bottleneck estimation is based on the packet pair approach described by Bolot [3]. 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 sender reports an application defined part (APP) 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 (R) is calculated as:

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

The probe packets can be ordinary data packets sent in a row. Using data packets as probe packets restricts the additional bandwidth required for the bottleneck probing to the increased size of the RTCP packets. The choice of running the measurement of the bottleneck bandwidth after the sending of each RTCP sender report or every few ones should be left to the application.

3.4.2 Scalability Enhancements for RTCP

For the sender to properly take the heterogeneity of the receivers into account when determining the number and sizes of layers to transmit, it needs feedback information representing the different rates calculated at the receivers. However, having all receivers sending feedback information to the sender would lead to scalability problems as the number of feedback packets would linearly increase with the number of receivers.

Therefore, we propose a mechanism we call partial suppression with which all possible rates a receiver can calculate are divided into intervals. That is, if the possible rates a receiver could calculate might vary between R_{min} and R_{max} we divide this range into subintervals $[R_{\text{min}}, R_1), [R_1, R_2), \cdots, [R_n, R_{\text{max}})$. After finishing the observation period the receiver calculates a theoretically TCP-friendly rate and determines in which subinterval this rate is found in. The receiver schedules now the transmission of the receiver report after some time period (T_{wait}). If a report form another receiver indicating a rate in the same subinterval was seen during this period the receiver suppresses the transmission of its own report.

For realizing efficient suppression, Nonnenmacher et al. [11] suggest using a truncated exponentially distributed timer in the interval $[0, T_{\text{rand}}]$ with the density of

$$T_{\text{wait}} = \begin{cases} \frac{1}{\exp^{\lambda} - 1} \times \frac{\lambda}{T_{\text{rand}}} \exp^{\frac{\lambda}{T_{\text{rand}}} z} & 0 \leq z < T_{\text{rand}} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

For ($\lambda = 10$) and ($T_{\text{rand}} = 5c$) with c as the delay between two receivers [11] shows analytically that for 10000 receivers less then 10 feedback messages are generated for each event the receivers are reporting on.

For dividing the possible rates into n subintervals we suggest using the following equation:

$$n = n_{\text{max}} - n_{\text{min}} \tag{3}$$
$$R_{\text{min}} = (1 + \rho)^{n_{\text{min}}} \tag{4}$$
$$R_{\text{max}} = (1 + \rho)^{n_{\text{max}}} \tag{5}$$

with ρ as the difference in percentage between two subsequent subintervals. For a possible rate range from 1 kb/s to 100 Mb/s, ($\rho = 0.1$) and the additional restriction that the difference between two subsequent subintervals should at least be 5 kb/s the number of subintervals is 90. For the case of partial suppression we would then expect around $(n * 10)$ receiver reports as a reaction to each sender report. Thus, if for each subinterval $[R_i, R_{i+1})$ there were a few thousand receivers that determine a rate $(R_i \leq R < R_{i+1})$ we would theoretically have around 900 feedback messages every T_{SR} for $(\lambda = 10)$ and $(T_{\text{rand}} = 5c)$.

Finally, while the sender reports are important for all receivers, the receiver reports are only of meaning to the sender and receivers of similar capacities, i.e., receivers that determine similar theoretical rates. Hence, the sender reports are sent on the basis layer[1]. The receivers, however, should send their reports only on the highest layer they are currently listening to. This avoids overloading receivers listening to the lower layers with the reports of the receivers listening also to higher layers.

3.5 TCP-Friendly Rate Adaptation

With MLDA each receiver determines the bandwidth share a TCP-connection traversing the same path between the receiver and the sender could utilize.

As obvious from Eqn. 1 the bandwidth share of a TCP connection depends on the loses and round trip delay this connection suffers. Hence, a TCP-friendly adaptation scheme needs to consider the loss and delay value in its bandwidth estimation. Here, losses can be determined by detecting the gaps in the sequence numbers of the incoming data packets.

3.5.1 Round Trip delay Estimation

The round trip delay between the sender and receiver (τ) can be determined as twice the difference between the timestamp of the incoming sender reports (T_{sender}) and the time the report arrives at the receiver as indicated by the receiver's local clock (T_{receiver}).

$$\frac{\tau}{2} = T_{\text{receiver}} - T_{\text{sender}} \tag{6}$$

A smoothed round trip delay (t_{RTT}) can then be estimated similar to the approach used with TCP [21]

$$t_{\text{RTT}} = \quad t_{\text{RTT}} + 0.125 \times (\tau - t_{\text{RTT}}) \tag{7}$$

However, such an approach requires some kind of synchronization of the clocks among the receivers and sender. Additionally, Paxon [14] shows through extensive measurements in the Internet, that the delay between two points might differ greatly on both directions. Thus Eqn.6 should be reformulated as

$$\frac{\tau}{2} + \delta = T_{\text{receiver}} - T_{\text{sender}} + \sigma \tag{8}$$

with δ as the difference $\frac{\tau}{2}$ and between the actual one way delay between the sender and receiver and is in the range of $(-\frac{\tau}{2} < \delta < \frac{\tau}{2})$. σ is the offset between the clocks of the sender and receiver due to their asynchronous behavior.

[1] Note that RTCP messages are usually sent on the same multicast address as the data packets but with a different port number

To get an estimation of the round trip delay (τ) we use an approach combining end-to-end measurements every T_τ and periodical one way measurements on intervals of ($T_{\text{ow}} \ll T_\tau$).

For the end-to-end measurements the MLDA receivers include in their receiver reports their local timestamp ($\dot{T}_{\text{receiver}}$) and the sender includes in its reports its timestamp (T_{sender}). Additionally, the sender reports include for each seen receiver report an entry indicating the identity of the reporting receiver, the time passed in between receiving the report and sending the sender report (T_{DLSR}) as well as the timestamp included in the receiver report ($\dot{T}_{\text{receiver}}$). With the end-to-end measurements we get an accurate value of τ and an estimation ($\theta : \theta = \sigma - \delta$) of the link asymmetry and of the offset between the clocks. θ can then be determined from the following equations:

$$\tau = T_{\text{receiver}} - \dot{T}_{\text{receiver}} - T_{\text{DLSR}} \tag{9}$$

$$\theta = T_{\text{receiver}} - T_{\text{sender}} - \frac{\tau}{2} \tag{10}$$

θ is then updated with each end-to-end measurement. In between two end-to-end measurements, the receiver updates its estimation of τ using θ and the timestamps of the periodically arriving sender reports as in Eqn. 8.

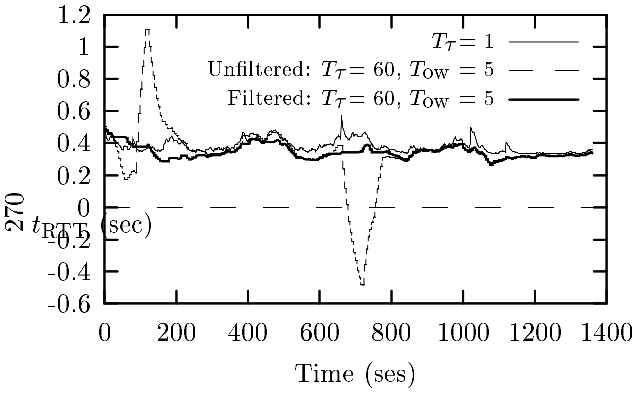


Figure 1: Delay measurement with MLDA

We have tested the accuracy of our delay measurement approach by running several measurement between donald(Germany) and ale(Berkeley), see Tab. 2. The results depicted in Fig. 1 show the estimated smoothed round trip delay (t_{RTT}) when using a round trip delay measurement every one second ($T_\tau = 1$) and for the case of running an end-to-end measurement every 60 seconds ($T_\tau = 60$) and updating t_{RTT} using one way measurements every 5 seconds ($T_{\text{ow}} = 5$). The initial round trip delay was set to 0.5 second. In general using our estimation mechanism resulted in values very close to those determined by doing an end-to-end measurement every second. However, in Fig. 1 we could identify three sources of problems:

1. As the system clocks of different hosts run at different speeds, a skew in the one way delay measurement exists that leads to an error in the calculation.

2. Due to some irregularities in the network or the end systems the end-to-end measurement might be wrong which would lead to wrong estimates of θ followed by wrong estimates of t_{RTT}. For example, the measurement done at time 660 seconds resulted in a high round trip delay. As this value was used for updating θ this lead to negative one way delay estimations and negative values of t_{RTT}.

3. Due to their inaccuracy, system clocks of computers are usually adjusted using a more accurate clock that is otherwise not directly accessible by the user[2]. This resulted in large jumps on the order of 1 second in the measured one way delay and in false τ values. This effect can be seen in the sudden increase in t_{RTT} at the 120th. second.

To filter out such irregularities we used the following approach:

End-to-end measurement: For the case that an end-to-end measurement resulted in a τ with

$$|\tau - t_{\mathrm{RTT}}| > t_{\mathrm{RTT}} \times \xi$$

then we save the value of the last calculated θ to θ_{old} and determine a new θ as in Eqn 10. Additionally, we save the previous value of τ to τ_{old}.

If using the newly calculated θ with the next N one way delay measurements delivers estimations of τ that are in the range of $[\tau_{\mathrm{old}} \times (1 \pm \gamma)]$ then the newly calculated value of θ is used until the next end-to-end measurement. Otherwise, the receiver uses θ_{old} until the next measurement after which a new θ is determined.

One way measurement: For the case the determined τ after a one way delay measurement fulfills the following inequality

$$|\tau - t_{\mathrm{RTT}}| > t_{\mathrm{RTT}} \times \xi \tag{11}$$

we save the newly determined τ to τ_{tmp}. Additionally, we calculate for each one way measurement a value δ as

$$\delta = T_{\mathrm{receiver}} - T_{\mathrm{sender}} \tag{12}$$

In case the inequality Eqn. 11 is fulfilled we determine a value ϕ as the difference between the value of δ calculated during the latest one way delay measurement and the previous value of δ.

If the next N one way delay measurements deliver estimations of τ in the range of $[\tau_{\mathrm{tmp}} \times (1 \pm \gamma)]$ then θ is increased by ϕ otherwise we assume the measurement that has delivered τ_{tmp} to be wrong and run the next measurements without adjusting θ.

Applying these filtering rules to the measurement results in Fig. 1 with N set to 3, ξ set to 0.5 and γ set to 0.5, lead to a smoothed round trip estimation with an error of maximally 20% from the one achieved with an end-to-end measurement every second.

[2] This adjustment occurs on the tested Solaris systems, when the system clock differs by more than 1 second from the more accurate one

Finally, note that the skew of the clocks plays a minor role here as its effects are offset by the end-to-end measurements. However, when increasing the interval between two end-to-end measurements, i.e., using larger values of T_τ, an approach for estimating the skew such as in [10] should additionally be used.

Having each receiver sending a report every T_τ would result in scalability problems even for the case of large values of T_τ on the order of a few minutes. On the other hand, depending only on the suppression mechanism might cause some receivers to never send a report and thus not be able to do an end-to-end measurement. Therefore we extend the scheme with the notion of local representatives. That is, receivers that send a report and make an end-to-end measurement announce to their neighboring receivers the determined θ. This is done by sending a special packet to the basic multicast layer with the time to live set to a small value to ensure that the packet will only reach geographically close members of the session. While we can assume that all neighboring receivers suffer from the same asymmetry to the sender, their clock offsets from the sender still differ. Thus, in the announced θ only the δ term is valid for the neighboring receivers. To get an estimation of the synchronization term (σ) of each receiver to the sender, the receivers start an end-to-end measurement of the round trip delay and a local θ (θ_{local}) to the announcing receiver. This is done similarly to the end-to-end measurement to the sender and calculation in Eqn. 10 but with the announcing receiver taking the role of the sender. Adding θ_{local} to the θ announced by that receiver all receivers get a valid estimation of their own θ with the sender.

3.5.2 Rate Calculation

As an example for an algorithm for calculating the TCP-friendly bandwidth share to utilize by non-TCP-controlled flows we use here an algorithm called loss-delay based adaptation algorithm (LDA) [19].

In contrast to our previous work in [19, 20], with MLDA the bandwidth share a flow can obtain between the sender and a receiver is estimated at the receiver instead of the sender.

LDA is an additive increase and multiplicative decrease algorithm with the addition and reduction values determined dynamically based on the current network situation and the bandwidth share a flow is already utilizing. During loss situations LDA estimates a flow's bandwidth share to be minimally the bandwidth share determined with Eqn. 1, i.e., the theoretical TCP-friendly bandwidth share determined using the TCP model. For the case of no losses the flow's share can be increased by a value that does not exceed the increase of the bandwidth share of a TCP connection with the same round trip delay and packet size.

In the detail, after receiving the mth. sender report (SR_m) the receivers start measuring the losses of the incoming data stream for a period of ($T_{\text{o}} \times x$) with T_{o} as the minimum measurement time needed to obtain usable loss information. x indicates the number of layers the receiver is already tuned to. After the measurement period each receiver (i) calculates the rate (r_{im}) that would be appropriate to use on the link connecting the sender to him. This calculation is done as follows:

No loss situation: In this case, the receiver (i) can increase its estimation of its TCP-friendly bandwidth share by an additive increase rate (A_{i}). To allow for a smooth increase of A_{i} and to allow flows of smaller bandwidth shares to faster increase their transmission rates than competing flows with higher shares, $A_{\rm i}$ is determined in dependence of the bandwidth share (r_i) the receiver is currently measuring relative to the bottleneck bandwidth (R_i) of the path connecting the sender to this receiver. Thus with an initial transmission rate of ($r_{\rm i_0}$), an initial additive increase value of $\dot{A}_{\rm i}$, $A_{\rm i}$ would evolve as follows:

$$A_{\rm i_1} = (2 - \frac{r_{\rm i_0}}{R_i}) \times \dot{A}_{\rm i} \quad (13)$$

$$A_{\rm i m} = (2 - \frac{r_{\rm i_{m\text{-}1}}}{R_i}) \times A_{\rm i_{m\text{-}1}} \quad (14)$$

Both $r_{\rm i_0}$ and $\dot{A}_{\rm i}$ are set by the user but should be kept small relative to the bottleneck bandwidth.

Finally, an RTP flow should not increase its bandwidth share faster than a TCP connection sharing the same link and having a similar round trip time. With the sender adjusting its transmission rate just before sending a sender report, i.e., every $T_{\rm SR}$ seconds and a round trip delay of (τ) a TCP connection would increase its transmission window by P packets with P set to

$$P = \sum_{q=0}^{T_{\rm SR}/\tau} q = \frac{(\frac{T_{\rm SR}}{\tau} + 1) \times \frac{T_{\rm SR}}{\tau}}{2} \quad (15)$$

with the window size being increased by one packet each round trip delay. Averaged over $T_{\rm SR}$, the RTP receiver should maximally increase its estimation of its bandwidth share by

$$A_{\rm i} = \frac{P}{T_{\rm SR}} \rightarrow \frac{\frac{T_{\rm SR}}{\tau} + 1}{2 \times \tau} \quad (16)$$

Finally, the receiver determines $r_{\rm i m}$ as

$$r_{\rm i m} = r_{\rm i_{m\text{-}1}} + A_{\rm i} \quad (17)$$

Loss situation: In this case the receiver reduces its estimation of the transmission rate the sender should be using on the path to him. $r_{\rm i}m$ is determined in this case as follows:

$$r_{\rm i m} = \max(r_{\rm i_{m\text{-}1}} \times (1 - \sqrt{l_{\rm i_{overall}}}), r_{\rm i_{TCP}}) \quad (18)$$

with $r_{\rm i_{m\text{-}1}}$ as the rate determined at receiver i after receiving the previous sender report ($SR_{\rm m\text{-}1}$), $r_{\rm i_{TCP}}$ determined using the TCP-model of Eqn. 1. Additionally, the increase factor (A) is reset to $\dot{A}$.

$l_{\rm i_{overall}}$ indicates here the average loss measured over all the layers receiver (i) is listening to. For the case of listening to x layers $l_{\rm i_{overall}}$ is set as follows:

$$l_{\rm i_{overall}} = \frac{l_{\rm i_1} \times R_{\rm L_1} + \cdots + l_{\rm i x} \times R_{\rm L x}}{\sum_{k=1}^{x} R_{\rm L_k}} \quad (19)$$

with $l_{\rm i}$k as the loss measured over layer k at receiver i and $R_{\rm L_k}$ as the rate of layer k as indicated in latest sender report ($SR_{\rm m}$).

3.6 Synchronous Join and Leave Actions

For the case of multicast, a data stream traverses a network node as long as at least one receiver behind this node is listening to this layer. Thus, the action of leaving a layer results in an overall rate reduction at this node only if all receivers behind this node leave this layer as well. Additionally, if two receivers joined different layers at the same time, the receiver joining the lower layer might observe losses that were not caused by the rate increase due to his join action but by the other receiver joining a higher layer.

To coordinate the actions of the different receivers we use the end of the observation periods as implicit synchronization points. That is, receivers can try to join the first enhancement layer after the end of the observation period of the basic layer (T_{o_0}). The second enhancement layer can then be joined or the receivers leave the first enhancement layer at the end of the observation period for the first enhancement layer, i.e., ($T_{\mathrm{o}_0} + T_{\mathrm{o}_1}$) after receiving the sender report.

Using synchronization points in itself is not novel. [25] already use this approach based on specially flagged packets. Note, however, that due to the heterogeneity of the network the sender reports (or the specially flagged packets) arrive at the receivers at different time points depending on the delay between the sender and receiver. To reduce the effects of this delay variation we extend the observation period for the basic layer by ($T_{\mathrm{sync}} = \frac{\tau_{\mathrm{max}} - \tau_{\mathrm{i}}}{2}$) with τ_{max} as the maximum seen round trip delay between the sender and the receivers in the multicast session and τ_{i} as the estimated round trip delay at receiver i. For estimating τ_{max} the receivers include their estimation of the round trip delay (t_{RTT}) to the sender in the receiver reports. The sender determines then based on the receiver reports the maximum round trip delay (τ_{max}) and includes this value in its reports. Note that as not all receivers send reports the τ_{max} estimation at the sender might not be completely correct. Additionally, as the t_{RTT} of the receivers are only estimations and the round trip delay tends to vary on short time scales this approach does not guarantee a perfect synchronization among the receivers but still manages to improve it compared to simply relying on the sender indications.

3.7 Reliability Issues

The dependency on the sender reports for initiating the adaptation actions of the receivers might lead to a deadlock situation. During overload periods the sender reports might get lost. However, without these reports the receivers do not start the observation periods and then possibly leave the higher layers due to the losses. Therefore, the congestion situation prevails and more sender reports might get lost. To avoid this situation, the receivers schedule a new observation period after a timeout of ($\sigma \times T_{\mathrm{SR}} : \sigma > 1$) with T_{SR} as the time period between sending two reports at the sender. Thus, if the sender report was lost, the receiver would start an observation period followed by joining or leaving a layer and sending a receiver report as was described above. In case a sender report was received before the timeout expires the scheduled actions are cancelled and new ones are scheduled.

3.8 Parameter Settings

In the description of MLDA we have used several parameters that control the temporal behavior of the algorithm. The sender transmits a report every $(T_{\text{SR}} + p)$, the receivers measure the behavior of a layer for T_{o} and schedule the transmission of a report in a period of $[0, T_{\text{rand}}]$.

The average period between sending two subsequent RTCP reports from an end system is usually 5 seconds. The work done for the case of unicast communication in Sec. 5 suggest that this value is large enough to maintain a low control overhead and yet sufficient to achieve efficient congestion control. We therefore set T_{SR} to 5 seconds and p arbitrarily to 0.5 seconds. To allow for a good degree of suppression we set T_{rand} to 1.5 seconds which is 5 times a typical half of the round trip delay between Europe and the States as measured between Berlin and Berkeley. Finally, note that for avoiding the loss multiplicity problem and for taking the adaptation decision based on the rates determined by all receivers, the receiver reports triggered by a sender report need to arrive at the sender before sending the next report. Therefore, the time left for the observation periods for all layers $(T_{\text{o}_{\text{all}}})$ can be determined as:

$$T_{\text{o}_{\text{all}}} = T_{\text{SR}} - p - T_{\text{rand}} - T_{\text{return}} - T_{\text{sync}}$$

with T_{return} as the time it takes the receiver reports to arrive at the sender and should be set at least to half the maximum round trip delay. With T_{return} and T_{sync} set arbitrarily to 0.5 seconds $T_{\text{o}_{\text{all}}}$ has a value of 2 seconds. After several simulations we decided to use an observation period of 0.5 seconds and we set the observation period of the basic layer to 1 second. Using smaller values resulted in inaccurate loss values and a highly oscillative adaptation behavior. This means that the scheme supports only one basic layer and two enhancement ones. Increasing T_{SR} to 6 seconds would allow for supporting two additional layers. While this naturally presents a limitation, using a larger number of layers would, however, increase the complexity of the receivers as they need to resynchronize the data from different layers. As the different layers might actually take different routes to the receiver, they might suffer from different delays. To reconstruct a data segment consisting of data packets sent over different layers the receiver would need to wait until all the different parts are received. This incurs additional delay and thus might reduce the overall perceived quality.

3.9 Compatibility and Implementation Issues

To allow compatibility with RTP receivers unaware of MLDA, the sender writes the additional fields for describing the layers and MLDA related parameters as an application specific part (APP) [17] of the RTCP control messages. Likewise, the receivers return their estimated TCP-friendly rate and round trip time as an additional application specific part of their RTCP messages. If the receivers do not support MLDA, the APP parts can be ignored. These receivers can still join the multicast session and receive the data of the basic layer. A receiver driven adaptation scheme such as [9] can be used to join higher layers.

To allow for dynamic changes of the adaptation parameters at the sender site the timing information such as T_{o} and T_{rand} should be included in the sender reports as well.

4 Simulative Performance Study of MLDA

To test the performance of MLDA in heterogeneous multicast environments we use the topology depicted in Fig. 2 with a multicast session consisting of a sender and 6 receivers connected to the sender over routers with different capacities. Each router is shared between the RTP stream and m TCP connections that have the same end-to-end propagation delay as that of the RTP sender/receiver pair. The sender transmits packets of 1 kbytes and each router is a random early drop (RED) [4] router. 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 that all flows receive the same loss ratio and avoids synchronization among the flows. Actually, the measurements in Sec. 5 and other simulation studies [18] suggest that the performance of MLDA is not affected by the used buffer management scheme. Here we use a maximum queuing delay of 0.15 seconds and set the maximum and minimum thresholds to 0.8 and 0.3 of the maximum buffer size. The router R0 works only as a distributer of data and incurs no losses or delays to the RTP data. The simulation is run for 800 seconds with the receiver connected to R1 joining the session after 300 seconds and leaving it again after 200 seconds. In our simulations we set the round trip propagation delay from the sender to each receiver and back to 100 msec. An accurate estimation of the bottleneck bandwidth is available at the receivers right after the start of the simulation. We additionally assume that the clocks of the sender and receivers are synchronized and that the round trip delay is measured accurately.

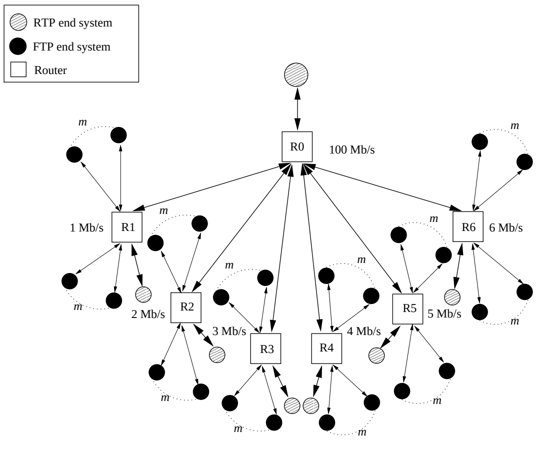
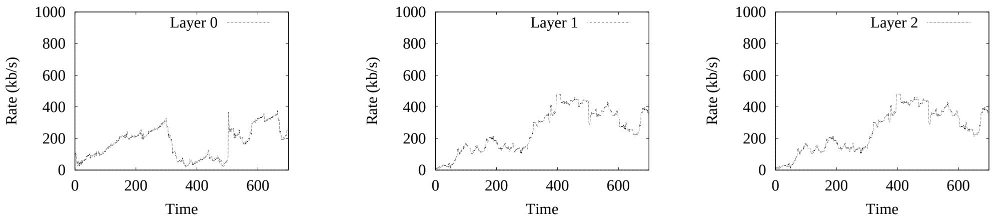


Figure 2: Multicast testing topology

Fig. 3 shows the distribution of bandwidth among the different layers for the case when each router is shared between an RTP flow and nine TCP connections. We notice that during the first 300 seconds the lowest layer has a bandwidth share of around 200 kb/s, i.e., the fair share for the receiver connected to R2, which has a capacity of 2 Mb/s. After the receiver connected to R1, which has a capacity of 1 Mb/s, joins the session the size of the basis layer is reduced down to the appropriate level of the new worst receiver, i.e., 100 kb/s. The sizes of the upper layers is increased to compensate the reduction in the basis layer. Thus the receivers listening to the higher layers are not affected by the reduction in the basis layer. After the receiver connected to R1 leaves the session at 500 seconds the size of the basis layer is increased again and, hence, improving the bandwidth share of the receivers listening only to the lower layer.



(a) Layer 0 (b) Layer 1 (c) Layer 2

Figure 3: Bandwidth distribution of the layers

Fig. 4 display the bandwidth distribution between the competing RTP and TCP flows at the different routers.

For the case of different numbers of competing TCP connections (m) Tab. 1 indicates that the ratio of the average bandwidth share of the RTP receivers to their theoretical one (r_{l_i}) is lower than one . The theoretical bandwidth share is determined as the available capacity at a router divided by the number of competing flows.

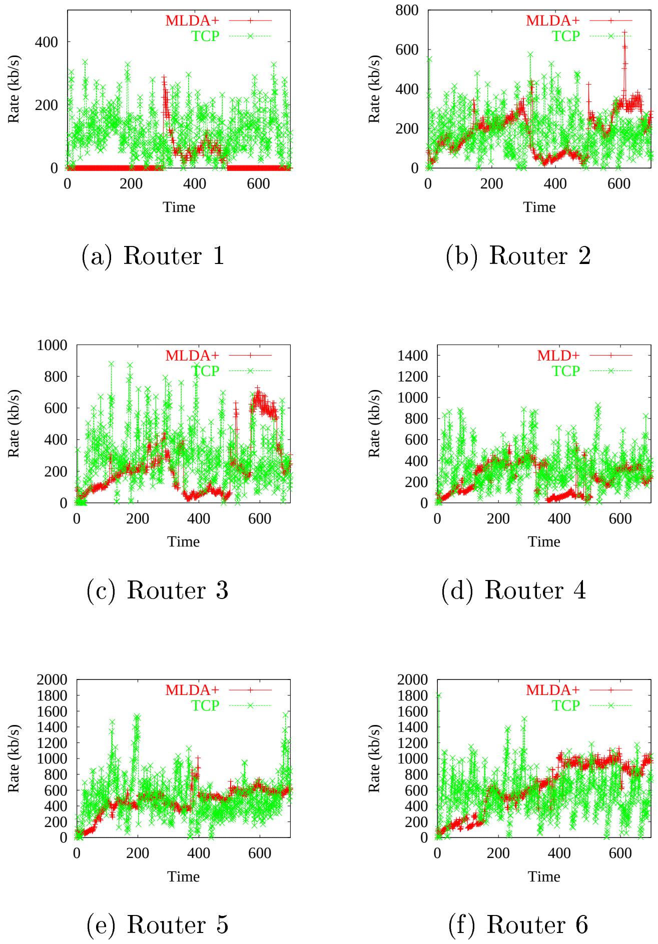
Both the results in Fig. 4 and Tab. 1 indicate that MLDA is actually rather conservative in its adaptation performance and allows competing TCP connections to acquire a higher bandwidth share than the MLDA flows. This results partially from the chosen number of layers and layer sizes. As we only used three layers and set the layer sizes in accordance with the maximum and minimum bandwidth shares ($r_{\max}$ and $r_{\min}$) receivers with shares of ($r : r_{\min} < r < r_{\max}$) will be restricted to the basis or the first enhancement layer and will only rarely benefit from their actual fair share. Improved fairness can only be reached using a larger number of layers [16] which, however, increases the complexity of the scheme as the end systems need to manage and resynchronize a larger number of layers.

Flows (m)	r_{l_2}	r_{l_3}	r_{l_4}	r_{l_5}	r_{l_6}
1	0.73	0.63	0.72	0.77	0.79
9	0.72	0.64	0.55	0.53	0.61

Table 1: Fairness index (F) and (r_{l_i}) with MLDA and m TCP connections on each link

5 Measurements of the TCP-Friendliness of MLDA

We tested the TCP-friendliness of MLDA by conducting unicast measurements over different parts of the Internet. Each measurement consisted of a host sending 10000 packets of 1 kbyte



(a) Router 1 (b) Router 2

(c) Router 3 (d) Router 4

(e) Router 5 (f) Router 6

Figure 4: Bandwidth distribution between TCP and MLDA+ at the different routers

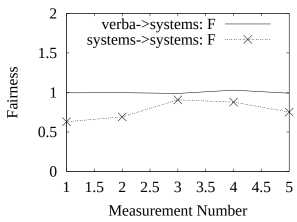
each over a TCP connection to some destination as fast as it can. Simultaneously, the same host sends the same amount of UDP packets to that destination with the transmission rate determined using LDA. Each measurement was done several times over different times of the day.

The host names and domains as well as their operating systems are listed in Tab. 2. The initial additive increase rate ($\dot{R}_{\mathrm{ai}}$) was set to 5 kb/s and the initial transmission rate of the UDP flows to 80 kb/s. We additionally limited the maximum transmission rate to 800 kb/sec. Similar to [15] the friendliness (F) of LDA is determined here as the goodput rate of the TCP connection divided by the goodput of the RTP stream.

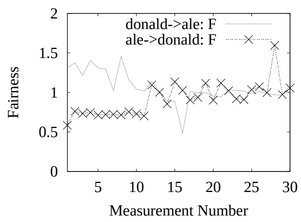
The links between verba and donald as well as verba and systems are rather lossy in both directions. Under these conditions the measured friendliness index (F) varies between 0.6 and 1.4 with most of the measured values in the range of 0.8 and 1.2 which are rather close to the optimal value of 1. The results depicted in Fig. 5(d) are, however, contradictory. In the direction form donald to systems we have a friendliness factor of around 0.8 on the average which means that the LDA controlled flow actually receives a smaller share of the

Host name	Domain	Operating System
donald	fokus.gmd.de	SunOS 5.5
verba	stu.neva.ru	SunOS 5.6
systems	seas.upenn.edu	SunOS 5.5
ale	icsi.berkeley.edu	SunOS 5.6

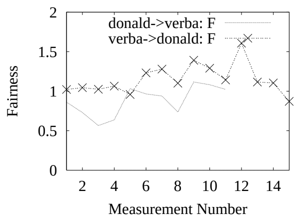
Table 2: Hosts used in the experimental tests



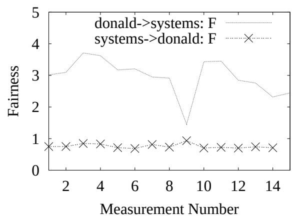
(a) systems⟺ verba



(b) donald⟺ ale



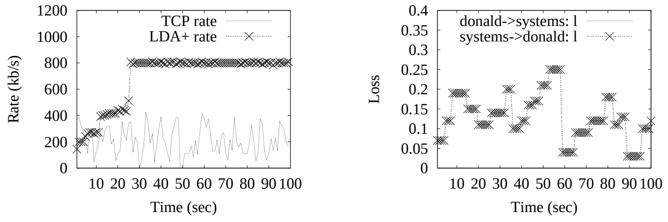
(c) donald⟺ verba



(d) donald⟺ systems

Figure 5: TCP friendliness (F) measured over the Internet

bandwidth than the competing TCP connection. The measurements on the opposite direction indicate, however, that the LDA controlled flow receives four times as much bandwidth as the competing TCP connection. Actually, the LDA controlled flow managed to achieve the maximum transmission rate and to stay at this level. These contradictory values resulted from the asymmetry of the Internet link between Europe and the States. Fig. 6 shows a detailed snapshot of the measurement labeled 2 in Fig. 5(d) and displays the transmission rate of both the TCP connection and LDA controlled flow and the losses measured over intervals of one second in both directions on the link connecting donald and systems. In Fig. 6(b) we notice that while in the direction from donald to systems no losses were observed during the entire measurement period, the traffic from systems to donald faced losses ranging from 5% up to 25% during the same period. This asymmetry leads to the well known ack compression problem [26]. For the case of a slower or lossy back link, TCP acknowledgments might arrive in clusters or might get lost. In the worst case, this might cause a timeout at the TCP sender which leads to a severe reduction of the congestion window and the sender transmitting packets that have already reached the receiver. In any case, the clustering or loss of acknowledgments can result in idle states at the TCP sender and thus a rate reduction.



(a) Bandwidth distribution from donald to systems

(b) UDP measured loss on both directions between donald and systems

Figure 6: Bandwidth distribution and losses measured between donald and systems

So while the transmission rate of UDP flows is adapted only to the losses on the way from the sender to the receiver, the transmission behavior of TCP connections or any other control scheme based on acknowledgment packets from the receivers such as [13, 15] depends also on the loss conditions on the direction from the receiver to the sender as well. Thus, in this situation setting the transmission rate of the UDP senders exactly to the equivalent TCP rate would be ineffective and might lead to underutilizing the network.

Fig. 7 shows the goodput, measured in intervals of 2 seconds, of the competing TCP and RTP streams during a period of 200 seconds of the measurement shown as point 18 in Fig. 5(b). LDA shows a less oscillatory behavior than TCP and has in general a comparable rate to that of TCP.

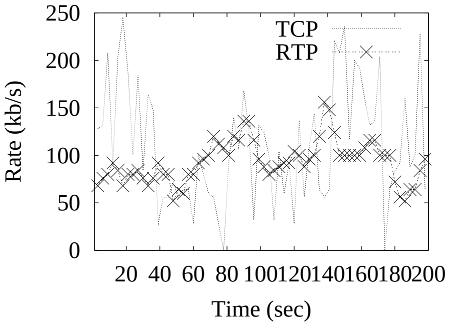


Figure 7: Temporal behavior of competing TCP and LDA streams

6 Summary and Future Work

In this paper we presented an adaptive rate control mechanism for unicast and multicast communication called MLDA. Our simulations and measurements suggest the efficiency of the scheme and its friendliness towards competing TCP traffic. In addition to the results presented here, we have run in [18] a large number of other simulations testing the adaptation behavior of LDA under varying conditions and different parameters that confirm the results presented here.

While MLDA was presented as a complete scheme it could be considered as a set of mechanisms that can be easily combined with other approaches. For example we could replace the rate calculation at the receivers (LDA) with some other such as [6]. The layering approach can be replaced by another one that might take better into account the heterogeneity of the network and the constraints imposed by the used coder or content. The here proposed mechanisms for providing the user with information about the heterogeneity of the network (partial suppression) or the delay measurement approach could also be used separately in some other congestion control approaches.

While the first results are promising we still need to run more simulations and do actual measurements to verify the performance of MLDA. Additionally, the scalability of the scheme needs to be investigated more in depth. The filtering approach for the delay measurements still requires further study to improve the accuracy and handle a wider range of possible error cases.

7 Acknowledgments

The RTP/RTCP simulation models were mainly implemented by Timur Friedman and improved by Frank Emanuel. The comments of Henning Sanneck and Henning Schulzrinne are gratefully acknowledged and were the basis for various aspects of this work. Special thanks Matthias Kranz for realizing the delay measurements, Andrey Vasilyev and Martin Reisslein for providing the the measurement end points.

References

[1] C. Albuquerque, B. Vickers, and T. Suda. An end-to-end source-adaptive multi-layered multicast (SAMM) algorithm. In 9th International Packet Video Workshop, New York, USA, Apr. 1999.

[2] S. Bhattacharyya, D. Towsley, and J. Kurose. The loss path multiplicity problem for multicast congestion control. In Proceedings of the Conference on Computer Communications (IEEE Infocom), New York, Mar. 1999.

[3] 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.

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

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

[6] M. Handely and S. Floyd. Strawman specification for TCP friendly reliable multicast congestion control, 1998. Note to the Internet Reliable Multicast Group mailing list.

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

[8] X. Li and M. H. Ammar. Bandwidth control for replicated-stream multicast video. 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 Societey.

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

[10] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In Proceedings of the Conference on Computer Communications (IEEE Infocom), New York, Mar. 1999.

[11] J. Nonnenmacher and E. W. Biersack. Optimal multicast feedback. In Proceedings of the Conference on Computer Communications (IEEE Infocom), San Francisco, USA, Mar. 1998.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In ACM SIGCOMM '98, Vancouver, Oct 1998.

[13] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A model based TCP-friendly rate control protocol. In Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999.

[14] V. Paxon. Measurements and Analysis of End-to-End Internet Dynamics. PhD thesis, Lawrence Berkeley National Laboratory, University of California, Berkeley, California, Apr. 1997.

[15] R. Rejaie, M. Handley, and D. Estrin. An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In Infocom'99, New York, March 1999. IEEE.

[16] D. Rubenstein, J. Kurose, and D. Towsley. The impact of multicast layering on network fairness. In Special Interest Group on Data Communication (SIGCOMM'99), Cambridge, USA, Aug. 1999.

[17] 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.

[18] D. Sisalem. A TCP-friendly adaptation scheme based on RTP. Technical report, GMD, Fokus, Germany, Apr. 1999.

[19] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Cambridge, England, July 1998.

[20] D. Sisalem and A. Wolisz. Towards TCP-friendly adaptive multimedia applications based on RTP. In Fourth IEEE Symposium on Computers and Communications (ISCC'99), Egypt, Jul 1999.

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

[22] W. Tan and A. Zakhor. Multicast transmission of scalable video using receiver-driven hierarchical FEC. In Packet Video Workshop 99, New York, Apr. 1999.

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

[24] 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.

[25] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In Proceedings of the Conference on Computer Communications (IEEE Infocom), San Francisco, USA, Mar. 1998.

[26] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way trffic. In SIGCOMM Symposium on Communications Architectures and Protocols, pages 133–147, Zrich, Switzerland, Sept. 1991. ACM. also in Computer Communication Review 21 (4), Sep. 1991.