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

Dorgham Sisalem
GMD FOKUS
Berlin, Germany
sisalem@fokus.gmd.de

Adam Wolisz*
GMD Fokus/TU Berlin
Berlin, Germany
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 general framework for achieving TCP-friendly congestion control called MLDA. Using MLDA, multimedia senders adjust their transmission rate in accordance with the network congestion state. For taking the heterogeneity of the Internet and the end systems into account, MLDA supports layered data transmission where the shape and number of the layers is determined dynamically based on feedback information generated by the receivers. Further, we discuss a measurement approach that allows receivers in large multicast sessions to estimate the round trip delay estimation to the sender in a scalable way. For exchanging control information between the sender and receivers we investigate the possibility of using the real time transport protocol (RTP) and discuss the required changes in order for RTP to support a more scalable and timely flow of feedback information from the receivers to the sender. Results obtained through simulations and measurements as well as comparisons to other congestion control schemes suggest that MLDA achieves TCP-friendly congestion control on the one hand and its ability to accommodate the needs of heterogeneous receivers on the other.**

I. Introduction

While congestion controlled TCP connections carrying time insensitive FTP or WWW traffic still constitute the major share of the Internet traffic today [1], 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 it 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 [2]. 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 [2].

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

- **Rate Adaptation:** Similar to the case of unicast congestion control, multicast congestion control should be TCP-friendly. Here, TCP-friendliness indicates that the bandwidth share consumed by the multicast flow on any traversed path resembles the bandwidth share of a TCP connection traversing the same path.

As the bandwidth share of a TCP connection depends on its round trip delay and losses, calculating a TCP-friendly bandwidth share involves determining losses and round trip times on all paths traversed by the multicast session. Hence, multicast congestion control schemes need to support scalable and accurate loss and delay measurement approaches.

* The work was partially done during the author's stay in ICSI Berkeley

• **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 [3] or referring to hierarchical data transmission [4], [5] 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 base layer improve the overall perceived quality of the data stream. Each layer is then sent on a different multicast group. 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. 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. Such approaches are particularly suitable for video communication using layered compression [6].

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). 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 by a competing TCP connection traversing the same path under the same loss and delay conditions. To satisfy the needs and capabilities of heterogeneous receivers, MLDA incorporates the concept of layered data transmission in its architecture. Adaptive schemes using layered data transmission usually assume statically set layer sizes. However, to better accommodate the actual heterogeneity of the receivers, MLDA senders periodically collect the information about the determined bandwidth shares at the receivers, adjust the sizes of the different layers they are transmitting based on these information and inform the receivers about the sizes and addresses of the different layers. The receivers can then determine the number of layers to join based on the information announced by the sender and their own estimation of the TCP-friendly rate they could be using.

MLDA is a hybrid sender and receiver-based adaptation scheme that combines on the one hand various well known concepts for multicast congestion control such as receiver-based rate calculation presented by Handley et al. [7], layered transmission [4] and dynamic layering presented by Sisalem et al. [8] into a unified congestion control architecture. On the other hand, MLDA provides novel solutions for round trip delay measurements, scalable feedback collection and a general framework for the cooperation between the sender and receivers that enables a seamless integration of these different concepts.

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 the exchange of control messages in a timely manner and to suppress the transmission of receiver information of similar content. Additionally, to get an estimation of the round trip delay between the sender and receivers we propose a simple measurement 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 in this work an algorithm we presented previously called the enhanced loss-delay based adaptation algorithm (LDA+) [9].

In Sec. II we present a brief overview of some related work in the area of TCP-friendly congestion control. Sec. III describes MLDA and the methods for estimating the network characteristics to be used for determining the TCP-friendly bandwidth share. In Sec. IV we discuss the issue of realizing MLDA using the real time transport protocol (RTP) [10] widely used for multimedia communication in the Internet. In Sec. IV-A, we investigate through simulations and measurements the performance of MLDA in a variety of settings and its ability to accommodate the needs of heterogeneous multicast receivers in a TCP-friendly manner.

II. Background and Related Work

Recently, there has been several proposals for TCP-friendly adaptation schemes that vary in their complexity, efficiency and goals.

Various congestion control schemes for UDP-based communication deploy an analytical model [11] of TCP for estimating the TCP-friendly bandwidth share of the UDP flow. With this model, the average bandwidth share of a

TCP connection (r_{TCP}) is determined as

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

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 Eqn. 1 for estimating the bandwidth share, requires exact knowledge of the round trip delay between the sender and receivers as well as the loss values on the paths connecting the sender to the receivers.

As an example for such an approach, Handley et al. present in [7] a scheme in which the receivers estimate using 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. The scheme does not currently provide for a scalable and accurate approach for measuring the round trip delay at the receivers and does not consider the case of heterogeneous receivers.

Bolot et al. [12] and Tan et al. [13] present schemes in which the sender transmits data in layers and the receivers calculate the rate appropriate for them using an analytical TCP model [14]. Based on this calculation they join the appropriate number of layers. Similar to [7], the issue of determining the round trip delay at the receivers is not addressed here. Additionally, in contrast to MLDA, these schemes only support layered transmission with the sizes and number of layers statically set by the sender.

In a different approach, MTCP [15] and RMTP [16] use a congestion window similar to that of TCP with the main difference that the window is only increased if all receivers acknowledge the reception of some packet. The increase and decrease actions of those protocols are similar to that of TCP. Such an approach adapts the transmission rate of the sender down to the capacity of the worst receiver of the multicast session and does not accommodate the needs of heterogeneous receivers. Additionally, MTCP and RMTP use a complex architecture for aggregating the receiver reports which introduces significant management overhead.

Vicisano et al. [5] 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. The scheme does not consider the round trip delay in its adaptation behavior and only supports statical layering of data.

The packet pair receiver-driven layered multicast (PLM) [17] is based on layered transmission and on the use of the packet pair approach to infer the bandwidth available at the bottleneck to decide which are the appropriate layers to join. To apply the packet pair approach for reliably estimating a flow's bandwidth share, the authors of PLM assume that all routers in the network deploy some kind of a fair queuing mechanisms that allocate each flow a fair bandwidth share.

Sisalem et al. present in [8] a first approach for incorporating sender-based rate adaptation with dynamic shaping of data layers.

Jiang et al. [18] present a scheme in which the sender transmits its data in a fixed base layer and a variable enhancement layer. Based on their measured losses, the receivers estimate their bandwidth share and report this to the sender. The transmission rate of the variable layer is then determined as to increase the satisfaction of most of the receivers.

Albuquerque et al. [19] describe an approach called source adaptive multi-layered multicast (SAMM), in which the receivers inform the sender about their desired transmission rate. To avoid feedback implosion, the feedback messages are not sent directly to the sender but to a representative that aggregates the information before forwarding it to the sender. In addition to the overhead introduced by having to maintain a list of representatives, SAMM requires substantial support from the network routers in order to drop data of higher layers during congestion periods as well as isolating the SAMM traffic from TCP traffic in order to achieve fair distribution.

III. Multicast Enhanced Loss-Delay Adaptation Algorithm (MLDA)

MLDA provides for a general framework for realizing congestion control in heterogeneous environments. The functionality of MLDA is realized on an end-to-end basis without requiring any support from the network routers beyond their capability of forwarding and routing multicast traffic.

The basic message exchange of MLDA is as follows:

1. The sender periodically transmits reports containing information about the sent layers.
2. After receiving a sender report each receiver ($j : j = 0, 1, \cdots, n$) measures the loss and delay of the incoming data for a period of time and determines a TCP-friendly bandwidth share (r_{j}) 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 receivers decide to join a higher layer, stay at or leave the current one.

4. Further, the receivers schedule the transmission of reports indicating their calculated bandwidth share after a random period of T_{wait}. Finally, if a report form another receiver with rate indication similar to r_{j} 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.

In the following subsections, see Sec. III-A and Sec. III-B, we first present the behavior of the sender and receivers and then describe in Sec. III-D different mechanisms for estimating the loss, delay and capacity characteristics of Internet links and an algorithm that uses this information for estimating the TCP-friendly bandwidth share of a flow. Note, that while the sender and receiver behavior description constitutes an integral part of the MLDA framework, the algorithms for estimating the network characteristics and the TCP-friendly bandwidth share are more generic in nature and can be easily replaced by other algorithms without altering the functionality of MLDA.

A. Sender Behavior

With MLDA the sender is responsible for adjusting its transmission behavior in accordance with the bandwidth available for the multicast session as estimated and reported by the receivers. The sender periodically polls feedback information from the receivers by sending a sender report at intervals of ($T_{\text{control}}+p$). p is a uniformly distributed random variable that assures that the reports from senders that start at the same time do not get synchronized. The sender includes in its reports information about the number of layers (y) it is transmitting, the rate ($R_{\text{L}_\text{k}} : k = 1, 2, \cdots, y$) of each layer ($L_{\text{k}}$) and the address on which each layer can be received on.

The receiver reports indicate the TCP-friendly bandwidth share estimated by the receivers to be available on the paths connecting the sender to them. The collected reports in between the sending of two sender reports are then used to adapt the sizes of the transmitted layers before sending the next sender report. To allow for fast reactions to changes in the network conditions the sender reduces the rate of the base layer whenever it receives a feedback message indicating a rate request lower than that used for the base layer.

A.1 Data Layering

Ideally, the sender would partition its data stream into a number of layers that satisfies the needs of all receivers. In a communication scenario with n receivers reporting transmission rates of ($r_1, \cdots, r_{\text{n}} : r_{\text{i-1}} < r_{\text{i}} < r_{\text{i+1}}$) the sender would then need to partition its data stream into substreams of ($r_2 - r_1, r_3 - r_2, \cdots, r_{\text{n-1}} - r_{\text{n}}$). Each receiver determining r_{j} as its available bandwidth share would then join x layers with ($r_{\text{j}} = \sum_{i=1}^{x} R_{\text{L}_\text{i}}$) for ($R_{\text{L}_1} = r_1$ and $R_{\text{L}_\text{i}} = r_{\text{i}} - r_{\text{i-1}}$).

However, using a large number of layers might result in drift problems [20], 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.

An example for an approach for determining the appropriate number and sizes of the layers would be to use data mining and clustering techniques [21] that allow the sender to summarize the receiver reports into a smaller set of representative values.

Note, that any 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 [19]. the only precondition for using such a dynamic layering approach is the availability of coders that can dynamically shape the number and sizes of layers [22].

When dividing data into layers, each layer should use its own ranges of sequence numbers. That is, while packets belonging to the same layer should have consecutive sequence numbers, this is not needed among packets belonging to different layers. This would allow the receivers to determine the loss rates of each layer. However, the receivers still need some means for resynchronizing the different layers into one data stream.

B. Receiver Behavior

Within the MLDA architecture, the receivers measure the characteristics of the paths connecting the sender to them, estimate the TCP-friendly bandwidth share they could consume, join the appropriate number of layers in accordance with their estimated bandwidth share and inform the sender about their estimated shares. In this section, the rules governing the join and leave actions of MLDA receivers are presented. Proposals for solving the issues of bandwidth estimation, path measurements and scalable feedback are presented in the next sections.

The join and leave actions of the receivers are triggered by the sender reports which are generated periodically in intervals of T_{control} by the sender.

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 (j) calculates the rate (r_{j}) that would be appropriate to use on the link connecting the sender to

him.

With $R_{\mathrm{L_k}}$ as the transmission rate on layer L_k the receiver can now take one of the following actions:

- $r_{\mathrm{j}} \geq \sum_{k=0}^{x+1} R_{\mathrm{L_k}}$**:** 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.
- $r_{\mathrm{j}} < \sum_{k=0}^{x} R_{\mathrm{L_k}}$**:** In this case, the rate of the incoming data stream is higher than the theoretically TCP-friendly share determined by some adaptation algorithm. The receiver must, thus, leave the highest layers it is currently receiving until the inequality ($r_{\mathrm{j}} \geq \sum_{k=0}^{x} R_{\mathrm{L_k}}$) is satisfied.
- $\sum_{k=0}^{x} R_{\mathrm{L_k}} < r_{\mathrm{j}} < \sum_{k=0}^{x+1} R_{\mathrm{L_k}}$**:** The receiver stays at the current level.

Finally, the receiver schedules the transmission of a report carrying the value of r_{i}.

The time passing between issuing a join request for a multicast layer and receiving the data for that layer is usually only the time needed for extending the multicast distribution tree towards the receiver. Hence, 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. After receiving a leave request, the network routers that forward the multicast data to the receiver that has issued the leave request need to make sure that the session is no longer requested by any other receiver before stopping forwarding the data on the links towards that receiver.

MLDA receivers directly determine the number of layers they can listen to. Hence, there is no need for observing the network situation after dropping a layer and taking the leave latency into account. Note also, that losses caused by a failed join operation by a receiver do not have any effects on the loss estimation of other receivers. That is, if some receivers are listening to x layers and share a part of the multicast tree they would have an observation period of ($T_{\mathrm{o}} \times x$). Only after the period of ($T_{\mathrm{o}} \times x$) would the receivers be allowed to join a higher layer. In case one of the receivers tried to join the next higher layer ($x+1$) and failed, the losses caused by the failed operation would not be included in the loss estimations of the other receivers who have already completed their observation period.

C. Scalable Feedback

In order to take the heterogeneity of the network and receivers into account in its adaptation decision the sender needs to collect feedback information representing all receivers.

Therefore, we propose a mechanism we call *partial suppression* with which all possible rates a receiver can calculate are divided into S 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 $[R_{\mathrm{min}}, R_1), [R_1, R_2), \cdots, [R_{\mathrm{S\text{-}1}}, R_{\mathrm{max}})$ with R_{min} and R_{max} as pre-defined constants. After finishing the observation period each receiver (j) calculates a theoretically TCP-friendly rate (r_{j}) 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. [23] suggest using a truncated exponentially distributed timer in the interval $[0, T_{\mathrm{rand}}]$ with the density of

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

For ($\lambda = 10$) and ($T_{\mathrm{rand}} = 5c$) with c as the delay between two receivers [23] 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 S subintervals we suggest using the following equation:

$$R_1 = R_{\mathrm{min}} \times (1+\rho) \quad (3)$$

$$R_{\mathrm{s}} = R_{\mathrm{s\text{-}1}} \times (1+\rho) \quad (4)$$

with ρ as the difference in percentage between two subsequent subintervals and ($s = 1, 2, \cdots, S$).

The total number of subintervals (S) can then be determined as follows:

$$R_{\mathrm{max}} = R_{\mathrm{min}} \times (1+\rho)^S \quad (5)$$

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 $[R_{\mathrm{s}}, R_{\mathrm{s+1}})$ there were a few thousand receivers that determine a rate ($r : R_{\mathrm{s}} \leq r < R_{\mathrm{s+1}}$) we would theoretically have around 900 feedback messages every T_{control} for ($\lambda = 10$) and ($T_{\mathrm{rand}} = 5c$).

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 base layer only. 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.

D. Receiver-Based Measurement of Path Characteristics

From Eqn. 1 we can see that for determining a TCP-friendly bandwidth share we need to take losses as well as delays on the links between the sender and receiver into account. Additionally, the receiver should never ask for a bandwidth share higher than the bottleneck rate, i.e., the capacity of the smallest router, on the path connecting the sender to the receiver.

In this part of the work, we present how the loss can be determined for the case of layered transmission as well as a novel approach for measuring the round trip delay.

D.1 Loss Estimation

By requesting that data packets carry sequence numbers the loss of packets can be recognized at the receivers by gaps in those numbers. Hence, the receivers can estimate the loss as the percentage of data packets to the number of packets sent by the sender during an observation period. The number of actually sent packets can be approximately indicated by the packet with the highest sequence number seen during the observation period.

To estimate an overall loss value (l) over all received layers the receiver measures the losses over each layer. For the case of listening to x layers l is determined as follows:

$$l = \frac{l_{\mathrm{L}_1} \times R_{\mathrm{L}_1} + \cdots + l_{\mathrm{Lx}} \times R_{\mathrm{Lx}}}{\sum_{k=1}^{x} R_{\mathrm{L_k}}} \qquad (6)$$

with $l_{\mathrm{L_k}}$ as the loss measured over layer k and $R_{\mathrm{L_k}}$ as the rate of layer k as indicated in the latest sender report.

D.2 Bottleneck Bandwidth Measurement

To estimate the maximum possible bandwidth share a flow can utilize one can refer to the packet pair approach first used by Bolot [24] for estimating the characteristics of Internet link. 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.

Hence, by sending probe packets at the access speed of the sender, the receiver can determine the bottleneck bandwidth (R) as follows:

$$R = \frac{\text{probe packet size}}{\text{gap between 2 probe packets}} \qquad (7)$$

The probe packets can be ordinary data packets sent in a row which would reduce the amount of required bandwidth for the bottleneck probing compared to the case when special packets are used. The choice of how often to run the measurement of the bottleneck bandwidth should be left to the application.

Due to losses of probe packets, network congestion or interference of other traffic Eqn. 7 might result in wrong estimations of the bottleneck bandwidth. Hence, the receivers need to deploy mechanisms such as [25], [26], [27] to filter out wrong estimates. Which filtering mechanism to use in the framework of MLDA is irrelevant, whereas schemes resulting in accurate estimations after short transient periods are to be favored.

D.3 Round Trip delay Estimation

The simplest approach for estimating the round trip delay between two systems is by sending a request message from one system to the other and having the other system acknowledging the request right away. The round trip delay can then be estimated as the time difference between sending the request and receiving the acknowledgment for it. While this end-to-end approach works fine for unicast communication, it does not scale for the case of multicast communication. For the case of multicast with a large number of receivers, having each receiver sending a request would overload the sender and the acknowledgments would also increase the load in the network. We therefore present in this section a novel approach that combines end-to-end measurements with one-way measurements.

With this approach the sender transmits in periods of T_{control} control messages containing timestamps (T_{sender}) indicating when these messages were sent. To avoid overloading the network and the sender with receiver control messages, the receivers might use suppression or some other approach for reducing the number of control messages. Hence, each receiver sends on the average a control message every ($T_\tau \ : \ T_{\text{control}} \ll T_\tau$). The combination of receiver control messages and sender messages can then be used by the receivers to accurately estimate the round trip delay. In addition to this end-to-end measurement approach the receivers can update their estimation of the round trip delay (τ) every T_{control} using the sender's timestamps (T_{sender}). With this one-way measurement approach the receivers estimate the round trip delay to the sender as twice the difference between the timestamp of an incoming sender report (T_{sender}) and the time the report

arrived at the receiver as indicated by the receiver's local clock (T_{receiver}).

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

A smoothed round trip delay (t_{RTT}) can then be estimated similar to the approach used with TCP [28]

$$t_{\text{RTT}} = \quad t_{\text{RTT}} + 0.125 \times (\tau - t_{\text{RTT}}) \tag{9}$$

However, for this one-way measurement approach to deliver accurate delay estimations some kind of synchronization of the clocks among the receivers and sender is required. Additionally, Paxon [26] shows through extensive measurements in the Internet, that the delay between two points might differ greatly in both directions. Thus, Eqn.8 should be reformulated as

$$\frac{\tau}{2} + \delta = T_{\text{receiver}} - T_{\text{sender}} + \sigma \tag{10}$$

with δ as the difference between $\frac{\tau}{2}$ and 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.

For the end-to-end measurements the receivers include in their control messages timestamps ($\dot{T}_{\text{receiver}}$) indicating when those messages were sent and the sender includes in its reports its timestamp (T_{sender}). Additionally, the sender reports include for each seen receiver message 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 ($\dot{T}_{\text{receiver}}$). The receivers indicated in the receiver list of the sender message can 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:

$$\tau = T_{\text{receiver}} - \dot{T}_{\text{receiver}} - T_{\text{DLSR}} \tag{11}$$

$$\theta = T_{\text{receiver}} - T_{\text{sender}} - \frac{\tau}{2} \tag{12}$$

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. 10.

We have tested the accuracy of this delay measurement approach by running several measurement between Berlin and Berkeley. The results depicted in Fig. III-D.3 show the estimated smoothed round trip delay (t_{RTT}) when using an end-to-end round trip delay measurement every one second

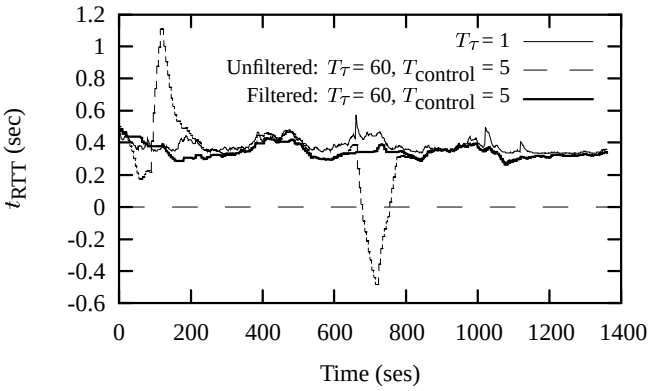


Fig. 1. Delay measurement with MLDA

($T_\tau = 1$) 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 ($T_{\text{control}} = 5$). The initial round trip delay was set to 0.5 seconds. In general using our estimation mechanism resulted in values very close to those determined by running an end-to-end measurement every second. However, in Fig. III-D.3 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[1]. 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-

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

end measurement resulted in a τ with

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

then we save the value of the last calculated θ to θ_{old} and determine a new θ as in Eqn 12. 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 $[\tau_{\text{old}} \times (1 \pm \gamma)]$ 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 each one-way measurement a value ϵ is calculated as

$$\epsilon = T_{\text{receiver}} - T_{\text{sender}} \tag{13}$$

For the case that the determined τ after a one-way delay measurement fulfills the following inequality

$$|\tau - t_{\text{RTT}}| > t_{\text{RTT}} \times \xi \tag{14}$$

we save the newly determined τ to τ_{tmp} and determine a value ϕ as the difference between the value of ϵ calculated during the last one-way delay measurement that did not fulfill Eqn. 13 and the ϵ determined for the first measurement that fulfilled Eqn. 13.
If the next N one-way delay measurements deliver estimations of τ in the range of $[\tau_{\text{tmp}} \times (1 \pm \gamma)]$ then θ is increased by ϕ otherwise the measurement that has delivered τ_{tmp} is assumed to be wrong and is ignored.

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

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 [29] 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, using a 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. 12 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.

E. 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}). Receivers listening to the first enhancement layer as well can join the second one leave the first enhancement layer at the end of the observation period for the first enhancement layer, i.e., ($T_{\text{o}_0} + T_{\text{o}_1}$) after receiving the sender report.

Using synchronization points in itself is not novel. Vicisano et al.[5] 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 ($T_{\text{sync}} = \frac{\tau_{\text{max}} - \tau_{\text{i}}}{2}$) 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 $\tau_{\max}$ 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.

F. 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 leave the higher layers in case the losses. Hence, in this case 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 ($\sigma \times T_{\text{control}} : \sigma > 1$) with T_{control} 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 maximally ($T_{\text{control}} \times \sigma$) seconds after receiving the last sender report. After the observation period, the receiver can join or leave a layer and schedule 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.

G. 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 ($T_{\text{control}} + p$), the receivers measure the behavior of a layer for T_{o} and schedule the transmission of a report in a period of $[0, T_{\text{rand}}]$.

Here we need to consider observation periods for different layers and accommodate possible delays in the leave actions. We therefore set T_{control} to 10 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 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:

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

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 7 seconds. After several simulations we decided to use an observation period of at least 1.5 seconds. Using smaller values resulted in inaccurate loss values and a highly oscillative adaptation behavior. This means that the scheme supports only up to around 5 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.

IV. Integration of RTP and MLDA

To allow for the cooperation of the sender and receivers in the MLDA framework different control information need to be exchanged. The sender periodically transmits a report with a description of the transmitted layers and some timing information. The receivers need also to send feedback information to the sender with timing and bandwidth information. The data packets themselves need to have sequence numbers in order for the receivers to detect losses as well some information that allow the receivers to resynchronize data of different layers into one data flow.

A protocol that already supports a large part of those requirements is the real time transport protocol (RTP) [10] widely used for multimedia communication in the Internet for carrying control information between the senders and receivers. RTP [10] 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.

RTP already includes sequence numbers in the data packet headers allowing the receivers to detect losses. Additionally, RTP includes timestamps in the data packets that enable the receivers to reestablish a timely sequence of the received data from different layers and hence resynchronizing different layers into one data stream.

However, to accommodate the different needs of MLDA some further additions are required for the integration of MLDA and RTP:

Time measurements: For the time measurements, the ses-

sion members already include timestamps indicating the report generation time in the RTCP reports. Additionally, each receiver indicates in its reports the timestamps of all sender reports seen since sending the last report, the identities of the senders and the time passed between receiving those reports and sending the receiver report. This allows the senders to estimate their round trip delay to the receivers.

As for the case of MLDA the receivers need to estimate their round trip delay to the sender as well. Hence, in this case, the senders need to include in their reports the timestamps of the received receiver reports since sending the last sender report, the identities of the reporting receivers and the time passed between receiving each report and generating the sender report.

Bottleneck measurement: To inform the receivers which data packets should be considered as probe packets, the RTCP sender can add to its 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 the packet numbered SEQ are sent at the access speed of the end system. The receiver can then use the time gaps between those packets for estimating the bottleneck.

Control information: The MLDA sender needs to inform the receivers about the sizes, addresses and number of transmitted layers as well as the maximum seen round trip delay. This information can be included in the sender RTCP reports as an application specific part. To allow the sender to dynamically change the adaptation parameters the timing information such as T_{o} and T_{rand} should be included in the sender reports as well. Similarly, the receivers can include their RTCP reports on their estimated bandwidth shares as application specific parts in their RTCP reports.

Control periods: As already mentioned in Sec. III the sender transmits its reports in fixed periods. After receiving a sender report the receivers schedule a feedback report and use the partial suppression approach for reducing the total number of reports.

The RTCP traffic is, however, 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. For large sessions this might result in very scarce reports not enough for achieving efficient adjustment of the sender behavior. Hence, this part of the RTCP specification needs to be adjusted to allow for a more scalable and timely flow of feedback information from the receivers to the sender.

If some receivers of the multicast data 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 [4] can be used to join higher layers.

A. Simulative Performance Investigation of MLDA

In this part of the work, we study the behavior of MLDA using simulations and measurements in the Internet. Here, we mainly concentrate on the TCP-friendliness of MLDA and its ability to accommodate the needs of heterogeneous receivers.

B. Estimation of a TCP-Friendly Bandwidth share

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 the enhanced loss-delay based adaptation algorithm (LDA+) [9].

In contrast to previous work in [30] and [9], 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 ($T_{\mathrm{o}} \times x$) 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 ($r_{\mathrm{i_m}}$) 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_{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 ($r_{\mathrm{i_0}}$), an initial additive increase value

of $\dot{A}_{\mathrm{i}}$, A_{i} would evolve as follows:

$$A_{\mathrm{i}_1} = \dot{A}_{\mathrm{i}} + (1 - \frac{r_{\mathrm{i}_0}}{R_{\mathrm{i}}}) \times \dot{A}_{\mathrm{i}} \quad (15)$$

$$A_{\mathrm{im}} = A_{\mathrm{i}_{\mathrm{m\text{-}1}}} + (1 - \frac{r_{\mathrm{i}_{\mathrm{m\text{-}1}}}}{R_{\mathrm{i}}}) \times A_{\mathrm{i}_{\mathrm{m\text{-}1}}} \quad (16)$$

Both r_{i_0} and $\dot{A}_{\mathrm{i}}$ are set by the user but should be kept small relative to the bottleneck bandwidth.
Finally, an adaptive 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_{SR} seconds and a round trip delay of (τ) a TCP connection would increase its transmission window by P packets with P set to

$$P = \sum_{q=0}^{T_{\mathrm{SR}}/\tau} q = \frac{(\frac{T_{\mathrm{SR}}}{\tau} + 1) \times \frac{T_{\mathrm{SR}}}{\tau}}{2} \quad (17)$$

with the window size being increased by one packet each round trip delay. With M as the packet size and averaged over T_{SR}, the receiver should maximally increase its estimation of its bandwidth share by

$$A_{\mathrm{i}_{\mathrm{m}}} = M \times \frac{P}{T_{\mathrm{SR}}} \rightarrow \frac{\frac{T_{\mathrm{SR}}}{\tau} + 1}{2 \times \tau} \quad (18)$$

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

$$r_{\mathrm{i}_{\mathrm{m}}} = r_{\mathrm{i}_{\mathrm{m\text{-}1}}} + A_{\mathrm{i}_{\mathrm{m}}} \quad (19)$$

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

$$r_{\mathrm{i}_{\mathrm{m}}} = \max(r_{\mathrm{i}_{\mathrm{m\text{-}1}}} \times (1 - \sqrt{l_{\mathrm{i}_{\mathrm{overall}}}}), r_{\mathrm{i}_{\mathrm{TCP}}}) \quad (20)$$

with $r_{\mathrm{i}_{\mathrm{m\text{-}1}}}$ as the rate determined at receiver i after receiving the previous sender report ($SR_{\mathrm{m\text{-}1}}$), $r_{\mathrm{i}_{\mathrm{TCP}}}$ determined using the TCP-model of Eqn. 1. Additionally, the increase factor (A) is reset to $\dot{A}$.
$l_{\mathrm{i}_{\mathrm{overall}}}$ indicates here the average loss measured over all the layers receiver (i) is listening to as described in Sec. III-D.1.

B.1 Simulation Model of MLDA

For the rate estimation part of MLDA we use in this study the loss-delay adaptation algorithm (LDA+) [9] as described in Sec. IV-B. While one could use some other scheme, we had already investigated LDA+ in various other studies and achieved good results in terms of stability and TCP-friendliness [9].

For dividing the data into K layers we refer here to a simple approach. 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 ($\frac{(\alpha \times r_{\mathrm{max}}) - r_{\mathrm{min}}}{K-1}$). ($\alpha : \alpha \leq 1$) 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}. In the simulations presented here, we set α to 0.9 ad K to 3. Note that while the chosen layering approach might have some impact on the fairness and stability of MLDA, the actual performance of MLDA does not depend on using a particular layering scheme.

The exchange of signaling information was realized in the simulation model using a model of RTP with the extensions described in Sec. IV. As a simplification, 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. This allows us to better investigate the aspects of TCP-friendliness and stability of the scheme without having to consider possible side effects of measurements errors.

Finally, we assume that the time passing between sending a leave request and this leave action actually taking effect is constant and is set to one second.

B.2 Performance of MLDA in Heterogeneous Multicast Environments

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 an MLDA stream and m TCP connections that have the same end-to-end propagation delay as that of the MLDA sender/receiver pair. The TCP connections are modeled as FTP flows that always have data to send and last for the entire simulation time. A TCP-Reno [28] model was used for simulating the congestion control behavior of TCP. The sender transmits packets of 1 kbytes and each router is a random early drop (RED) [31] 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. V and other sim-

ulation studies [32] 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 data. In our simulations we set the round trip propagation delay between the sender and the receivers to 200 msec.

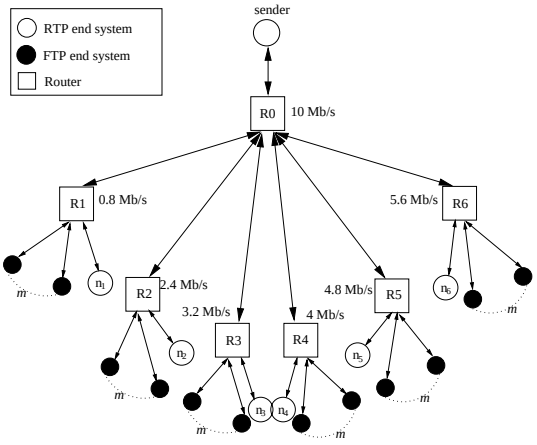
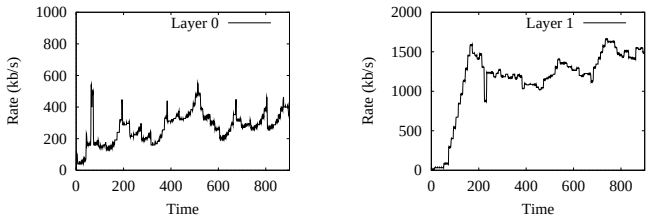


Fig. 2. Multicast testing topology

Fig. 3 shows the distribution of bandwidth among the different layers for the case when each router is shared between an MLDA flow and a TCP connection. We can observe that the rate of the basic layer, see Fig. 3(a), is adjusted in accordance with the capacity of receiver n_0.

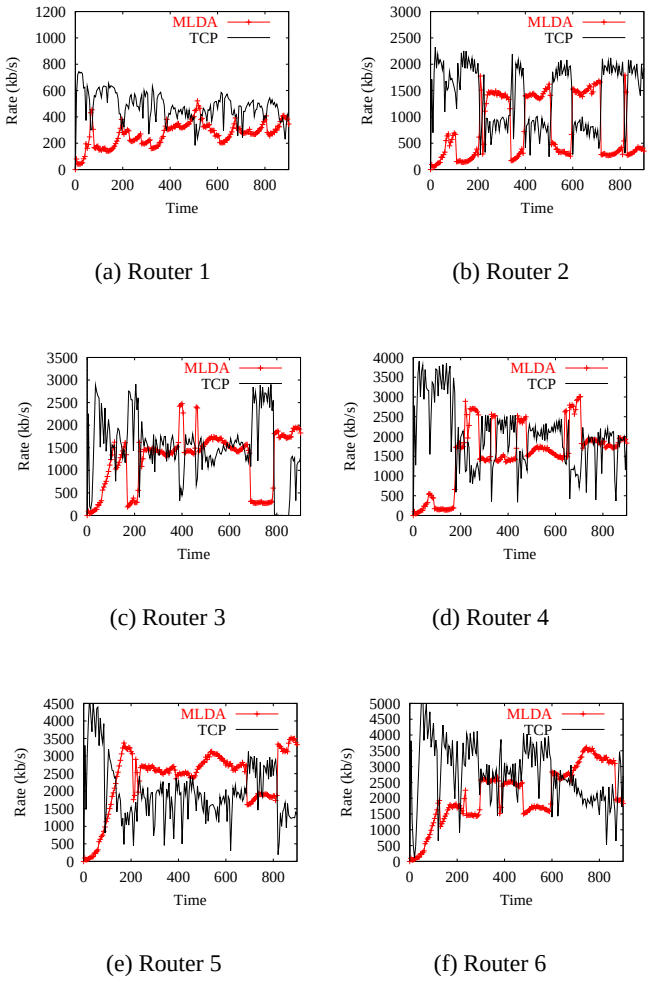


(a) Layer 0 (b) Layer 1, 2

Fig. 3. Bandwidth distribution of the layers

Fig. 4 displays the bandwidth distribution between the competing MLDA and TCP flows at the different routers.

For the case of different numbers of competing TCP connections (m) Tab. I indicates that the ratio of the average bandwidth share of the MLDA receivers to the average bandwidth share of the competing TCP connections at each router (R_n) varies around one. Both the results in Fig. 4 and Tab. I indicate that MLDA is in general fair towards the competing TCP connections and manages to satisfy the capabilities of the heterogeneous receivers. While



(a) Router 1 (b) Router 2

(c) Router 3 (d) Router 4

(e) Router 5 (f) Router 6

Fig. 4. Bandwidth distribution between TCP and MLDA at the different routers

the bandwidth shares of the receivers with the lower bandwidth (n_1 and n_2) is restricted by the adaptation scheme, receivers that have a TCP-friendly bandwidth share that is larger than the sum of the first x layers but below the sum of the $x+1$ layers will oscillate between layer x and $x+1$. Depending on the chosen layering approach and number of layers this might result in temporary unfair bandwidth distribution. Improved fairness and smaller oscillations can only be reached using a larger number of layers [33] which, however, increases the complexity of the scheme as the end systems need to manage and resynchronize a larger number of layers.

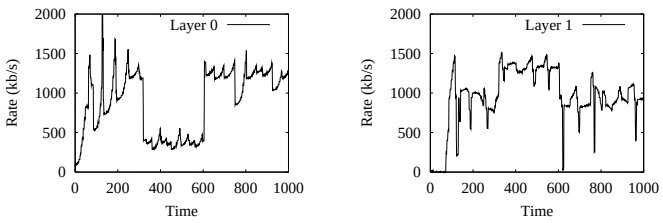
B.3 Performance of MLDA in Dynamical Environments

In this section, we investigate the behavior of MLDA in a dynamical receiver setting, i.e., a setting with the receivers joining and leaving the multicast session during the simulation time. For this purpose, we repeat the simula-

Flows (m)	R_1	R_2	R_3	R_4	R_5	R_6
1	0.66	0.70	1.03	1.04	1.5	0.95
8	0.50	0.90	0.85	1.09	0.92	1.02

TABLE I
RATIO OF AVERAGE BANDWIDTH SHARE OF THE MLDA FLOW TO THE AVERAGE SHARE OF COMPETING TCP CONNECTIONS

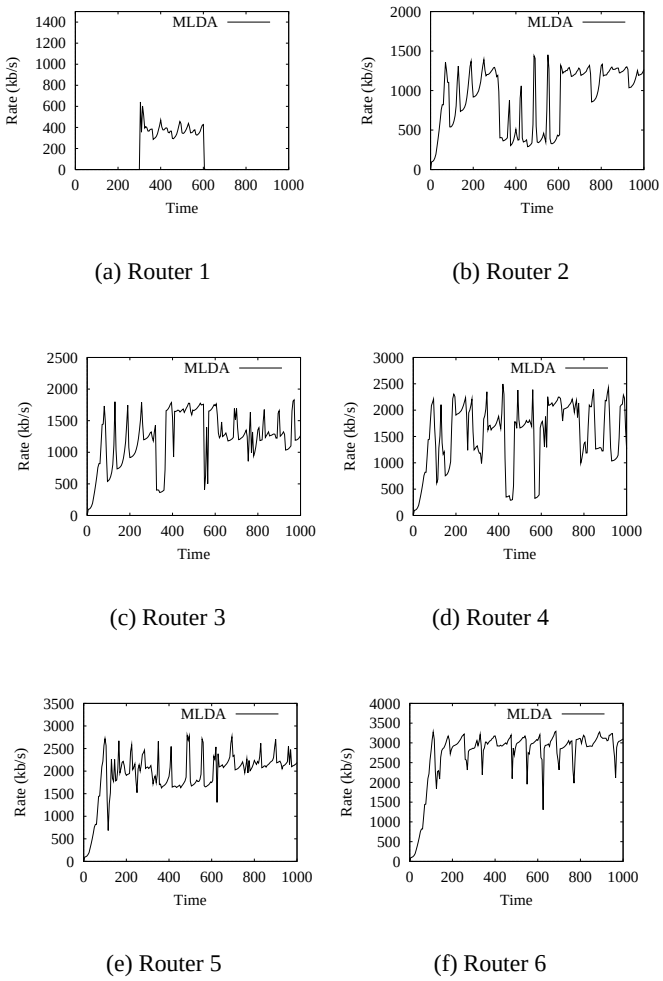
tions of Sec. IV-B.2 but with receiver (n_1) joining the multicast session at time 300 seconds and leaving it at time 600 seconds. Each router is shared between the MLDA flow and uncorrelated background traffic which consumes maximally half of the router's capacity. The background traffic is modulated as the aggregate of 100 on-off processes with the on period lasting for the time needed to carry a number of packets drawn from a Pareto distribution with the factor of 1.1 and a mean of 20 packets and the off period lasting for a time drawn from a Pareto distribution with a factor of 1.8 and a mean of 0.5 seconds [34].



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

Fig. 5. Bandwidth distribution of the layers

As Fig. 5 depicts, during the first 300 seconds the bandwidth share of the lowest layer is increased up to around 1.2 Mb/s which is the bandwidth share of receiver n_2 which constitute the worst receiver in this case. After receiver n_1 joins the session the size of the base layer is reduced down to the appropriate level of the new worst receiver, i.e., 400 kb/s. The sizes of the upper layers is increased to compensate the reduction in the base layer. Thus the receivers listening to the higher layers are not affected by the reduction in the base layer. The share of receiver (n_6) as depicted in Fig. 6(f) is not changed due to the join and leave actions and is comparable to the results achieved in Sec. IV-B.2, see Fig. 4(f). After receiver n_1 leaves the session at 600 seconds the size of the base layer is increased again and, hence, improving the bandwidth share of the receivers listening only to the lower layer.



(a) Router 1 (b) Router 2

(c) Router 3 (d) Router 4

(e) Router 5 (f) Router 6

Fig. 6. Bandwidth share of MLDA at the different routers

B.4 Performance of MLDA in Multicast Environments with Shared Links

In the previous section, see Sec. IV-B.2 and Sec. IV-B.3, we only considered multicast distribution trees without any shared links among the receivers. This was beneficial for investigating the TCP-friendliness of MLDA and its ability to accommodate heterogeneous receivers without having to consider the effects of interactions between different receivers, traversing multiple routers and different round trip delays among the receivers.

Fig. 7 depicts another configuration of a multicast tree that is shared among different receivers with each link of the tree having a different capacity. Similar to Fig. 2, RED routers are used. Each router is shared between the MLDA flow and uncorrelated background traffic which consumes maximally half of the router's capacity.

The bandwidth distribution among the layers, see Fig. 8, accommodates in this case the capacities of the receivers such that receiver n_5 manages to get a bandwidth share

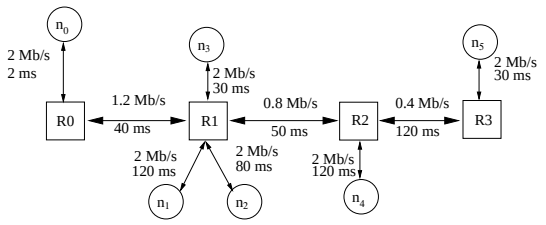
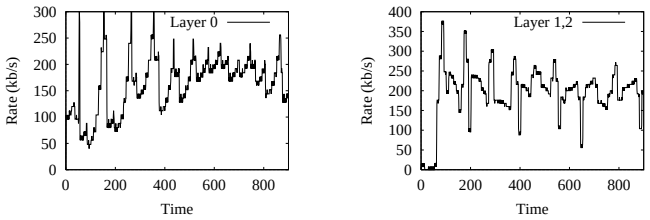


Fig. 7. Testing topology of shared multicast trees

of around 170 kb/s and the receivers connected to the first router (R_1) get a bandwidth share of 540 kb/s.



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

Fig. 8. Bandwidth distribution of the layers

As Fig. 9 suggests that receivers connected to the same router over links with similar bandwidth capacities but with different round trip delays receive identical bandwidth shares and stay synchronized throughout the simulation time. Also, notice that receiver (n_4) oscillates between the first two layers which might cause losses in the stream forwarded further towards receiver n_5. However, due to the synchronization mechanisms of MLDA those losses are ignored at receiver n_5 which does not include the losses caused of the failed join operations of receiver n_4 into its loss estimations.

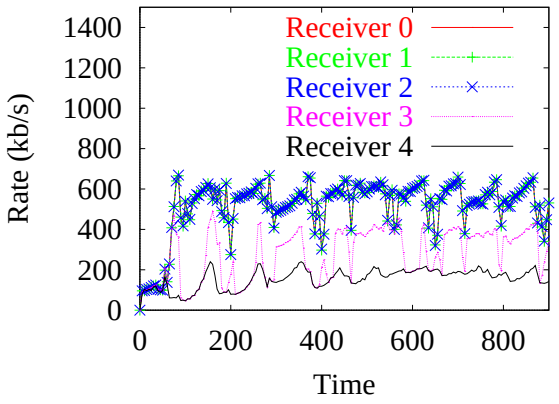


Fig. 9. Bandwidth distribution among the receivers

V. Measurements of the TCP-Friendliness of MLDA

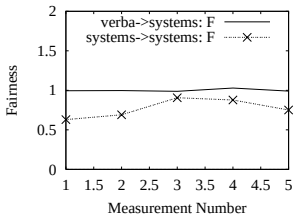
We tested the TCP-friendliness of MLDA by conducting measurements over different parts of the Internet. Each measurement consisted of a host sending 10000 packets of 1 kbyte each over a TCP connection to some destination as fast as it can. Simultaneously, the same host sends the same amount of UDP packets to that destination with the transmission rate determined using LDA+. Each measurement was done several times over different times of the day. To estimate the average bottleneck bandwidth incorrect estimates are filtered out using a similar approach to that deployed in the BPROBE tool [27]. That is, similar estimates are clustered into intervals and the average of the interval with the highest number of estimates is the chosen. As there was only one receiver in the test scenario, the receiver was able to frequently send receiver reports and measure the round trip delay on an end-to-end basis leading to rather accurate delay estimations.

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

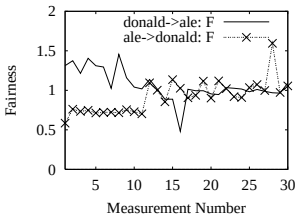
TABLE II
Hosts used in the experimental tests

The host names and domains as well as their operating systems are listed in Tab. II. The initial additive increase rate ($\dot{R}_{\text{ai}}$) 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 [35] the friendliness (F) of LDA+ is determined here as the goodput rate of the TCP connection divided by the goodput of the MLDA stream.

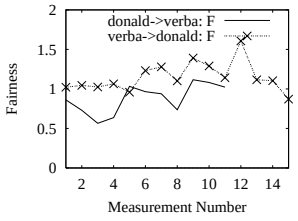
The links between *verba* and *donald* as well as *verba* and *systems* are rather lossy in both directions. Under these conditions the measured friendliness index (F) varies between 0.6 and 1.4 with most of the measured values in the range of 0.8 and 1.2 which are rather close to the optimal value of 1. The results depicted in Fig. 10(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 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



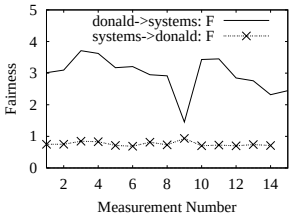
(a) *systems* to *verba*



(b) *donald* to *ale*

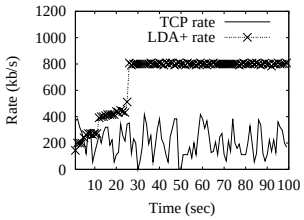


(c) *donald* to *verba*

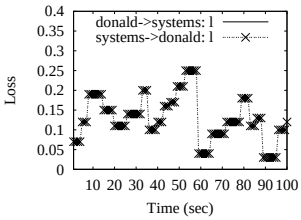


(d) *donald* to *systems*

Fig. 10. TCP friendliness (F) measured over the Internet



(a) Bandwidth distribution from *donald* to *systems*



(b) Measured loss on both directions between *donald* and *systems*

Fig. 11. Bandwidth distribution and losses measured between *donald* and *systems*

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. 11 shows a detailed snapshot of the measurement labeled 2 in Fig. 10(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. 11(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 [36]. 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.

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 [37], [35] 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. 12 shows the goodput, measured in intervals of 2 seconds, of the competing TCP and MLDA streams during a period of 200 seconds of the measurement shown as point 18 in Fig. 10(b). LDA+ shows a less oscillatory behavior than TCP and has in general a comparable rate to that of TCP.

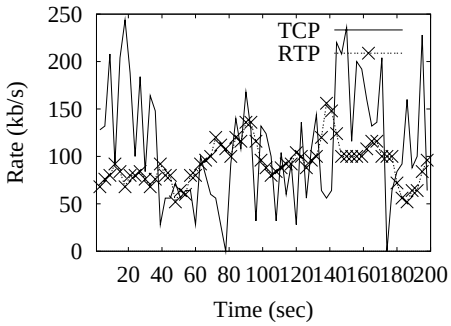


Fig. 12. Temporal behavior of competing TCP and LDA streams

VI. Comparison of the Performance of MLDA to Other Congestion Control Schemes

In this part of the work, we compare between the performance of MLDA and a row of recent proposals for TCP-friendly congestion control schemes for multicast communication. Due to the large number of such proposals we restrict our comparisons to only a few that combine TCP-friendly adaptation with layered data transmission. For comparing the schemes, we picked out some representative test cases as were described in the papers presenting the to be compared algorithms. We re-simulated those cases in our simulation environment and compared the achieved results using MLDA with the results achieved by the other schemes as were reported by their authors. This approach

reduces possible errors in the comparisons due to misinterpretations or wrong implementations of the algorithms.

A. Comparison of MLDA and PLM

The packet pair receiver-driven layered multicast (PLM) [17] is based on layered transmission and on the use of the packet pair approach to infer the bandwidth available at the bottleneck to decide which are the appropriate layers to join. PLM is receiver driven, i.e., congestion control is achieved through the join and leave actions of the receivers. The packet pair approach is used usually to estimate the bandwidth of the bottleneck router on the path between the sender and receiver and not in the bandwidth share of the transmitted flow. To apply the packet pair approach for reliably estimating a flow's bandwidth share, the authors of PLM assume that all routers in the network deploy some kind of a fair queuing mechanism that allocates each flow a fair bandwidth share. Only under this assumption, which is currently invalid in the Internet, is it possible to use PLM for congestion control.

With PLM the sender periodically sends a pair of its data packets as a burst to infer the bandwidth share of the flow. The receivers use the gaps between the specially marked packet pairs to estimate their bandwidth share. Based on the estimated share they determine the number of data layers they can receive.

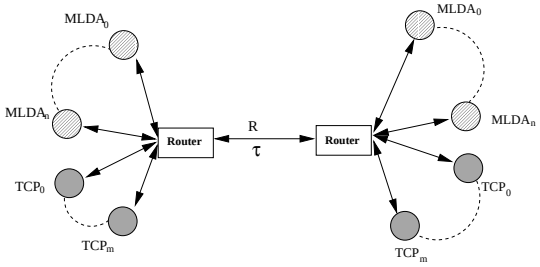
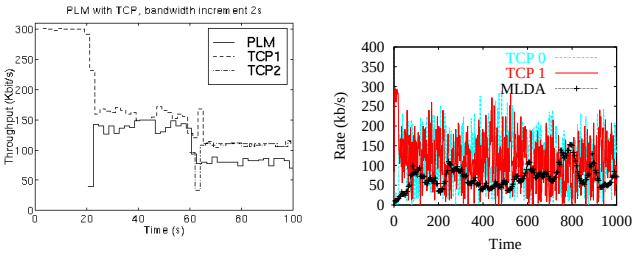


Fig. 13. Testing topology

For comparing the TCP-friendliness of PLM and MLDA we use the topology depicted in Fig. 13 with a bottleneck link shared between two TCP ($m = 2$) connections and an MLDA flow ($n = 1$). The bottleneck link has a bandwidth of 300 kb/s and a delay of 0.02 seconds. For reducing the effects of synchronization and ensuring a fair distribution of the losses we use a RED router with a maximum buffering delay of 0.2 seconds and the maximum and minimum thresholds set to 0.3 and 0.8. The initial transmission rate of the MLDA flow was set to 100 kb/s and the packet size was set to 500 bytes. The first TCP connection starts at time 0 seconds and the second one 60 seconds later. The adaptive flow starts at time 20 seconds. For PLM, a packet pair was sent each second and each flow had a queue size of 20 packets. Additionally, the sender transmitted 17 layers each of 20 kb/s. This topology was taken from [17] Fig.[2] whereas in [17] a router with fair queuing was used.



(a) PLM (b) MLDA

Fig. 14. Bandwidth distribution with MLDA and PLM

The results presented in Fig. 14 describe the bandwidth distribution between the TCP and MLDA flows when using PLM and MLDA. Measured over the entire simulation time, both approaches achieve similar average bandwidth distributions with the adaptive flow receiving around 80 kb/s and the TCP flows having a share of 110 kb/s. However, looking at the temporal behavior of the flows in Fig. 14 it can be observed that with PLM the flows show hardly any oscillations and have rather constant bandwidth shares. With MLDA on the contrary, the TCP flows oscillate between 0 kb/s and 250 kb/s. While the MLDA flow itself shows a less oscillatory behavior than TCP it still shows a variance of $\pm 50\%$ of its average bandwidth share. With PLM the receivers always know their exact bandwidth share and as the number of flows in the network is constant the bandwidth shares of the flows is constant as well. Hence, this stable behavior of the flows with PLM was to be expected.

Note that PLM not only assumes a fair queuing network but also that the packet pair approach always results in correct estimations of the bandwidth share. However, even in a fair queuing network, the packet pair approach might result in under or overestimation of the bandwidth share. That is, due to losses of probe packets, network congestion or interference of other traffic this approach might result in wrong estimations of the bandwidth. Hence, the receivers need to deploy mechanisms such as [25], [26], [27] to filter out wrong estimates. As such effects are difficult to simulate, the results presented in [17] for PLM are to be considered rather optimistic. Due to the effects of losses and traffic interference a higher degree of oscillations can be expected in a more realistic environment.

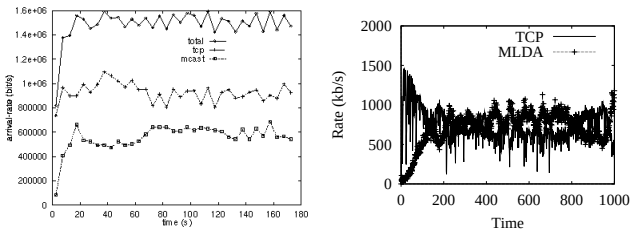
Finally, For the case of fair queuing networks deploying the packet pair approach with MLDA for the bandwidth es-

timation part would improve the performance of MLDA in terms of stability and maintain its flexibility in heterogeneous multicast environments.

B. Comparison of MLDA and RLC

Vicisano et al. present in [5] a receiver-driven layered control scheme (RLC) for realizing TCP-friendly congestion control. With RLC, the sender divides its data into layers and sends them on different multicast sessions. To test the availability of resources, the sender periodically generates short bursts of packets followed by an equally long relaxation period in which no packets are sent. For the duration of the bursts the consumed bandwidth by the flow is doubled. Hence, if the packets of the burst are not lost then this indicates the availability of resources. After receiving a packet burst, the receivers can join a higher layer if the burst was lossless otherwise they remain at their current subscription level. The receivers might leave a layer at any time if losses were measured.

In [5] a simulation topology is used similar to Fig. 13 but with the round trip delay (τ) set to 0.42 seconds and the bottleneck bandwidth (R) set to 1.5 Mb/s. The link is shared between 8 TCP connections and 8 adaptive flows and the packet size is set to 1024 bytes. Fig. 15(a) suggests that RLC shows a very conservative behavior under these conditions and the RLC flow receives only around 60% of the bandwidth share consumed by the TCP connection. The behavior of MLDA in this case resembles the behavior of TCP to a greater extent with the MLDA flow receiving around 90% of the bandwidth consumed by the TCP connection, see Fig. 15(b).



(a) RLC (b) MLDA

Fig. 15. Bandwidth distribution with RLC and MLDA

In a second test, we compare the behavior of RLC and MLDA in a multicast tree with shared links, see Fig. 16. This topology is similar to that described in Sec. IV-B.4 but with different delay and bandwidth parameters. Also in this case, the routers are shared between the adaptive flow and uncorrelated traffic.

As Fig. 17 describes, receivers connected to the same

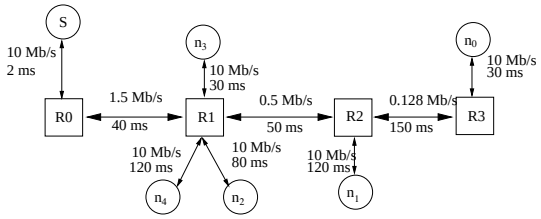
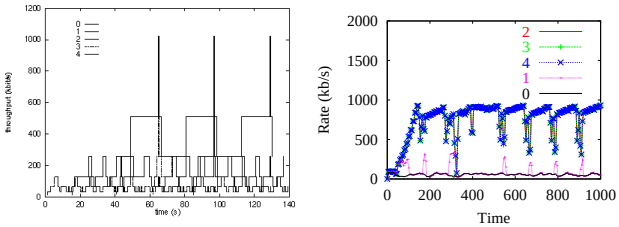


Fig. 16. Testing topology of shared multicast trees

bottleneck receive identical shares and stay synchronized throughout the simulation time with both RLC and MLDA. However, with RLC the maximum bandwidth share of the receivers (n_2, n_3, n_4) is restricted by the static nature of the transmitted layers. That is while the background traffic only consumes half of the 1.5 Mb/s, the receivers (n_2, n_3, n_4) can only utilize up to 500 kb/s instead of 750 kb/s. With MLDA the sizes of the layers is shaped in accordance with the available resources and the receivers (n_2, n_3, n_4) receive on the average a bandwidth share of around 750 kb/s. Hence, while MLDA introduces a higher complexity due to the exchange of control messages between the sender and receivers it is more capable of accommodating the needs of heterogeneous receivers than RLC.



(a) RLC (b) MLDA

Fig. 17. Behavior of RLC and MLDA in a shared multicast tree

C. Comparison of MLDA and IRFMC

Jiang et al. [18] present a scheme called inter-receiver fair multicast communication in which the sender transmits its data in a fixed base layer and a variable enhancement layer. Based on their measured losses, the receivers estimate their bandwidth share and report this to the sender. The transmission rate of the variable layer is then determined as to increase the satisfaction of most of the receivers.

Receivers determine their appropriate bandwidth share by measuring the losses of the incoming flow. In case no losses were observed the receivers increase their estima-

tions of their bandwidth share by ($\frac{r_{\text{current}}}{l_{\text{tol}}}$) with r_{current} as the current transmission rate of the session and l_{tol} as the loss that can be tolerated by the receiver. In case of losses, the receivers reduce their bandwidth estimation by a reduction factor intended to be similar to TCP's rate reduction in face of losses.

To compare the behavior of IRFMC with MLDA we chose a test topology depicted in Fig. 18 as was described in [18].

The type of the TCP connection is Tahoe-TCP [28] and it starts at 80 seconds and stops transmitting data at 250 seconds. The routers in the topology use the drop tail buffer management and have a buffer of 20 packets. The multicast flow starts transmitting data with a rate of 200 kb/s.

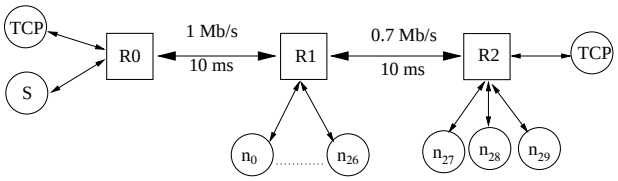
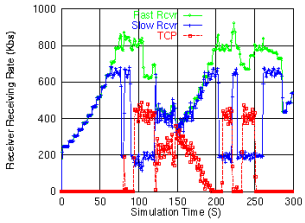
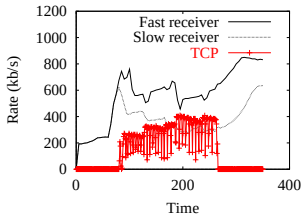


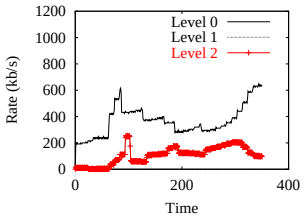
Fig. 18. Testing topology for MLDA and IRFMC



(a) RLC



(b) MLDA: Receivers



(c) MLDA: Levels

Fig. 19. Bandwidth distribution with IRFMC and MLDA

As Fig. 19(b) depicts, with MLDA the sender starts increasing the transmission rate of its base layer until it reaches the capacity of router R2. To further utilize the resources still available at router R1, the sender starts increasing the transmission rate of the enhancement layers. When the TCP connection starts transmitting data, the sender reduces the transmission rate of the base layer to allow for fair bandwidth distribution between the TCP connection and the slow receivers with each of the TCP and MLDA flows receiving a bandwidth share of 350 kb/s at router R2. Receivers connected to the faster router get a bandwidth share of around 600 kb/s during the active time of the TCP connection.

With IRFMC, the TCP connection achieves an average bandwidth share of 220 kb/s [18]. However, from Fig. 19(a) we can see that the TCP connection receives a bandwidth share of around 400 kb/s in the first half of its life time. In the second half its share is reduced considerably and it only receives a share of the bandwidth when the slower receivers temporarily leave the higher layer.

VII. Summary and Future Work

In this paper, we presented an adaptive rate control mechanism for multicast communication called MLDA. We have tested MLDA under various topologies with various parameters and compared the behavior of MLDA to a range of recently proposed TCP-friendly congestion control schemes. 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 [32] a large number of other simulations testing the adaptation behavior of LDA+ under varying conditions and different parameters that confirm the results presented here.

The results of comparing MLDA to other congestion control schemes, suggest that while the bandwidth distribution with MLDA is not as stable as that achieved with network-supported scheme such as [17] it is nevertheless still TCP-friendly. Compared to simple schemes that do not require the exchange of control information or the measurement of round trip delays, MLDA achieves a more TCP-friendly bandwidth distribution and is better suited to accommodate the needs of heterogeneous receivers.

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 [7]. 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.

VIII. 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 measurement end points.

References

[1] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, pp. –, November/December 1997.

[2] Sally Floyd and Kevin Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, Aug. 1999.

[3] Xue Li and Mostafa 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.

[4] Steven McCanne, Van Jacobson, and Martin Vetterli, "Receiver-driven layered multicast," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Palo Alto, California, Aug. 1996.

[5] Lorenzo Vicisano, Luigi Rizzo, and Jon 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.

[6] Steven McCanne, Martin Vetterli, and Van Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 6, Aug. 1997.

[7] Mark Handely and Sally Floyd, "Strawman specification for TCP friendly reliable multicast congestion control," 1998, Note to the Internet Reliable Multicast Group mailing list.

[8] Dorgham Sisalem and Frank Emanuel, "QoS control using adaptive layered data transmission," in *IEEE Multimedia Systems Conference'98*, Austin, TX, USA, June 1998.

[9] Dorgham Sisalem and Adam Wolisz, "Towards TCP-friendly adaptive multimedia applications based on RTP," in *Fourth IEEE Symposium on Computers and Communications (ISCC'99)*, Egypt, July 1999.

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

[11] 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.

[12] Thierry Turletti, Sacha Fosse Prisis, and Jean-Chrysostome Bolot, "Experiments with a layered transmission scheme over the Internet," Rapport de recherche 3296, INRIA, Nov. 1997.

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

[14] Sally Floyd and Fall Kevin, "Router mechanisms to support end-to-end congestion control," Tech. Rep., LBL-Berkeley, Feb. 1997.

[15] Injong Rhee, Nallathambi Balaguru, and George N. Rouskas, "MTCP: Scalable TCP-like congestion control for reliable multicast," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, USA, Mar. 1999.

[16] S. Paul, K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, April 1997.

[17] A. Legout and E. W. Biersack, "Fast convergence for cumulative layered multicast transmission scheme," Tech. Rep., Eurecom, Sophia-Antipolis, France, Oct. 1999, under submission.

[18] Tianji Jiang, Ellen Zegura, and Mostafa Ammar, "Inter-receiver fair multicast communication over the internet," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, New Jersey, June 1999.

[19] Celio Albuquerque, Brett Vickers, and Tatsuya Suda, "An end-to-end source-adaptive multi-layered multicast (SAMM) algorithm," in *9th International Packet Video Workshop*, New York, USA, Apr. 1999.

[20] Uwe Horn and Bernd Girod, "Scalable video coding for the internet,," in *8th Joint European Networking Conference*, Edinburgh, England, May 1997.

[21] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Int. Conf. VLDB*, Santiago, Chile, Sept. 1994.

[22] E. Miloslavsky and A. Zakhor, "Constant PSNR rate control for layered video compressing using matching pursuits," in *Proceedings of the International Conference on Image Processing*, Kobe, Japan, Oct. 1999.

[23] J. Nonnenmacher and Ernst W. Biersack, "Optimal multicast feedback," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, USA, Mar. 1998.

[24] Jean-Chrysostome Bolot, "End-to-end packet delay and loss behavior in the Internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, Deepinder P. Sidhu, Ed., San Francisco, California, Sept. 1993, ACM, pp. 289–298, also in *Computer Communication Review* 23 (4), Oct. 1992.

[25] Kevin Lai and Mary Baker, "Measuring bandwidth," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, USA, Mar. 1999.

[26] Vern Paxon, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, Lawrence Berkeley National Laboratory, University of California, Berkeley, California, Apr. 1997.

[27] Robert L. Carter and Mark E. Crovella, "Measuring bottleneck link speed in packet-switched networks," Tech. Rep. BU-CS-96006, Computer Science Departement, Boston University, Mar. 1996.

[28] W. Richard Stevens, *TCP/IP illustrated: the protocols*, vol. 1, Addison-Wesley, Reading, Massachusetts, 1994.

[29] Sue B. Moon, Paul Skelly, and Don 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.

[30] Dorgham Sisalem and Henning 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.

[31] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

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

[33] Dan Rubenstein, Jim Kurose, and Don Towsley, "The impact of multicast layering on network fairness," in *Special Inter-*

est Group on Data Communication (SIGCOMM'99), Cambridge, USA, Aug. 1999.

[34] Kihong Park, Gitae Kim, and Mark Corvella, "On the relationship between file sizes, transport protocols and self-similar network traffic," in *International Conference on Network Protocols (ICNP)*, Columbus, Ohio, Oct 1996.

[35] Reza Rejaie, Mark Handley, and Deborah Estrin, "An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Infocom'99*, New York, March 1999, IEEE.

[36] Lixia Zhang, Scott Shenker, and David 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*, Z rich, Switzerland, Sept. 1991, ACM, pp. 133–147, also in *Computer Communication Review* 21 (4), Sep. 1991.

[37] 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.