Abstract—In this paper, we present an end-to-end adaptation scheme, called the enhanced loss-delay based adaptation algorithm (LDA+), for regulating the transmission behavior of multimedia senders in accordance with the network congestion state. LDA+ uses the real-time transport protocol (RTP) for collecting loss and delay statistics which are then used for adjusting the transmission behavior of the senders in a manner similar to TCP connections suffering from similar losses and delays. Performance results collected using simulations and measurements over the Internet suggest the efficiency of LDA+ in terms of network utilization, congestion avoidance and fairness towards competing TCP connections.

I. INTRODUCTION AND MOTIVATION

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 introducing additional delays to the carried data due to retransmissions as is the case with TCP. Additionally, as UDP does not require the receivers to send acknowledgments for received data, UDP is well suited for multicast communication. However, deploying non-congestion controlled UDP in the Internet on a large scale might result in extreme unfairness towards competing TCP traffic as TCP-traffic reacts to congestion situations by reducing its bandwidth consumption and UDP doesn’t. 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 with 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 over the entire life time of the flow and not at every time point [2].

In this paper, we describe a new scheme called the loss-delay based adaptation algorithm (LDA+), that adapts the transmission rate of UDP-based multimedia flows to the congestion situation in the network in a TCP-friendly manner. Basically, LDA+ regulates the transmission rate of a sender based on end-to-end feedback information about losses, delays and the bandwidth capacity measured by the receiver. With no observed losses, the sender can increase its transmission rate additively otherwise it needs to reduce it multiplicatively.

The work presented here is an extension of our previous work presented in [3]. We have updated our work based on more recent proposals for analytical models of TCP [4], improved the used approach for dynamically determining the additive increase rate and finally the new algorithm avoids using parameters that had to be statically set by the user in the older version.

II. BACKGROUND AND RELATED WORK

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

Rejaie et al. present in [5] an adaptation scheme called rate adaptation protocol (RAP). Just as with TCP, sent packets are acknowledged by the receivers with losses indicated either by gaps in the sequence numbers of the acknowledged packets or timeouts. Using the acknowledgment packets the sender can estimate the round trip delay. If no losses were detected, the sender can periodically increase its transmission rate additively as a function of the estimated round trip delay. After detecting a loss the rate is multiplicatively reduced by half in a similar manner to TCP.

Padhye et al. [4] present an analytical model for the average bandwidth share of a TCP connection ($r_{TCP}$)

\[ r_{TCP} = \frac{M}{t_{RTT} \sqrt{\frac{2 DL}{3}} + t_{out} \min \left( 1, 3 \sqrt{\frac{3 DL}{8}} \right) l (1 + 32^2)} \]  

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

Using this model Padhye et al. [6] present a scheme in which the sender estimates the round trip delay and losses based on the receiver’s acknowledgments. In case of losses, the sender restricts its transmission rate to the equivalent TCP rate calculated using Eqn. 1 otherwise the rate is doubled. While the scheme behaves in a TCP-friendly manner during loss phases, its increase behavior during underload situations might result in un-
equal bandwidth distribution due to the possibility of increasing the transmission rate faster than a competing TCP connection.

Additionally, various schemes have been proposed for the case of multicast communication that combine either the additive increase and multiplicative decrease adaptation approach or the analytical based approach with layered data transmission for accommodating heterogeneous receivers [7], [8].

III. The Enhanced Loss-Delay Based Adaptation Algorithm (LDA+)

The enhanced loss-delay based adaptation algorithm (LDA+) is a sender based adaptation scheme. It relies on the real time transport protocol (RTP) [9] for feedback information about the losses at the receiver and round trip time. Based on this information the sender adjusts its transmission rate in accordance with the network congestion state.

A. Estimation 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 sender should not increase its transmission rate above the bottleneck rate of the link, i.e., the bandwidth of the smallest router on the path connecting the sender to the receiver.

LDA+ uses the real time transport protocol (RTP) for transporting control information between the sender and receiver. RTP 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 called RTCP, each member of a communication session periodically sends control reports to all other members containing information about sent and received data. Additionally, the end systems might include in their reports an application specific part (APP) intended for experimental use. The RTCP traffic is scaled with the data traffic so that it makes up a certain percentage of the data rate (usually 5%) with a minimum interval of 5 seconds between sending 2 RTCP messages. RTCP messages already include information about the losses and delays noticed in the network. Losses are estimated at the receiver by counting the gaps in the sequence numbers included in the RTP data header. For estimating the round trip delay, the sender includes in its control packets a timestamp and the receiver includes in its control packets information about the time passed between receiving the last sender report and sending this receiver report ($T_{pass}$). Thereby, the sender can estimate the round trip delay as the time passed between sending its report and receiving the receiver report minus $T_{pass}$. Additionally, we enhanced RTP with the ability to estimate the bottleneck bandwidth of a connection based on the packet pair approach [10].

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

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

The probe packets can be ordinary data packets sent in a row. Using data packets as probe packets restricts the additional bandwidth required for the bottleneck probing to the increased size of the RTCP packets. Due to losses of probe packets, network congestion or interference of other traffic Eqn. 2 might result in wrong estimations of the bottleneck bandwidth. Hence, the receivers need to deploy mechanisms such as [11], [12] to filter out wrong estimates. Which filtering mechanism to use in the framework of LDA+ is irrelevant, whereas schemes resulting in accurate estimations after short transient periods are to be favored.

B. Rate Adjustment with LDA+

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 $m$th. receiver report the sender estimates the bandwidth share ($r_m$) it should be using as follows:

No loss situation: In this case, the sender can increase its estimation of its TCP-friendly bandwidth share by an additive increase rate ($A$). To allow for a smooth increase of $A$ and to allow flows of smaller bandwidth shares to faster increase their transmission rates than competing flows with higher shares, $A$ is determined in dependence of the bandwidth share ($r_{m-1}$) the sender is currently consuming relative to the bottleneck bandwidth ($R$) of the path connecting the sender to the receiver. Thus with an initial transmission rate of ($r_0$), an initial additive increase value of $A$, $A$ would evolve as follows:

$$A_{\text{add}}_1 = (2 - \frac{r_0}{R}) \times A$$

$$A_{\text{add}}_m = (2 - \frac{r_{m-1}}{R}) \times A_{m-1} \quad (4)$$

Both $r_0$ and $A$ are set by the user but should be kept small relative to the bottleneck bandwidth.

To limit the rate increase maximally to the bottleneck bandwidth a second value of $A$ is additionally determined that converges to 0 as the bandwidth share of the flow converges to the bottleneck bandwidth. One function that fulfills this requirement is the exponential function in the form of

$$A_{\text{exp}}_m = (1 - \exp^{-(1 - \frac{r_{m-1}}{R})}) \times r_{m-1} \quad (5)$$
Finally, an RTP flow should not increase its bandwidth share faster than a TCP connection sharing the same link. With an average value of \( T \) seconds between the reception of two receiver reports and a round trip delay of \( \tau \), a TCP connection would increase its transmission window by \( P \) packets with \( P \) set to

\[
P = \sum_{p=0}^{T/\tau} p = \frac{(T + 1) \times T}{2}
\]

with the window size being increased by one packet each round trip delay. Averaged over \( T \), the RTP receiver should maximally increase its estimation of its bandwidth share by

\[
A_{TCPm} = \frac{P}{T} \rightarrow \frac{T + 1}{2 \times \tau}
\]

The additive increase value \( A_m \) is then set to

\[
A_m = \min(A_{addm}, A_{expm}, A_{TCPm})
\]

Finally, the sender determines \( r_m \) as

\[
r_m = r_{m-1} + A_m
\]

**Loss situation:** In this case the sender reduces its estimation of the transmission rate and \( r_m \) is determined in this case as follows:

\[
r_m = \max(r_{m-1} \times (1 - \sqrt{\tau}), r_{TCP})
\]

with \( r_{m-1} \) as the rate currently used by the sender, \( r_{TCP} \) determined using the TCP-model of Eqn. 1. Additionally, the increase factor \( (A) \) is reset to \( A \).

**IV. PERFORMANCE TESTING OF LDA+**

For testing the performance of LDA+ we use a simple topology, see of Fig. 1, consisting of a bottlenecked link connecting \( m \) RTP-based senders and receivers, \( n \) TCP-based FTP connections and \( k \) TCP connections carrying WWW traffic. The TCP connections are based on the Reno TCP specifications with fast retransmission and fast recovery algorithms [13]. The LDA+ and FTP sources are modeled as greedy sources that always have data to send at the highest possible rate as allowed by the congestion control mechanism during the entire simulation time. The WWW servers are modeled as 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 [14].

The link has a capacity of 10 Mb/s and a propagation delay of \( \tau \). The bottlenecked router (Router1) can introduce a maximal additional delay of 0.1 seconds due to the buffering of the data. The second router (Router2) serves only as a distributor of the data to the end systems and does not introduce losses or delays to the transported data.

Each simulation is run for 1000 seconds whereas the first 200 seconds are considered as a transient phase and are not taken into account in the here presented results.

Fig. 1. Adaptation performance testing topology

Similar to [5], the TCP-friendliness \( F \) of LDA+ is determined as

\[
F = \frac{r_{RTP}}{r_{TCP}}
\]

with \( r_{RTP} \) as the goodput of the RTP connection and \( r_{TCP} \) as the goodput of the TCP connection.

The packet size was set to 1000 bytes, \( \hat{A} \) to 5 kb/s and all the RTP connections start at the same time with an initial transmission rate \( (r_0) \) of 10 packets/sec.

**A. Competing LDA+ and FTP Flows**

In this section, we investigate the TCP-friendliness of LDA+ for the case of FIFO as well as random early drop (RED) [2] routers. Here, we only consider the interaction between LDA+ and FTP flows that always have to data to send for the entire simulation time. A RED gateway detects incipient congestion by computing the average queue size. When the average queue size exceeds a preset minimum threshold the router drops each incoming packet with some probability. Exceeding a second maximum threshold leads to dropping all arriving packets. This approach not only keeps the average queue length low but ensures fairness and avoids synchronization effects. Based on results achieved in [2] the minimum drop threshold was set to 0.3 of the routers buffer and the maximum one to 0.8 of the routers buffer. The queuing weight \( (w_j) \) was set to 0.002 and the maximum drop probability \( (P_d) \) was set to 0.02.

<table>
<thead>
<tr>
<th>( \tau )</th>
<th>RED</th>
<th>( N = 27 )</th>
<th>( N = 64 )</th>
<th>( N = 81 )</th>
<th>FIFO</th>
<th>( N = 27 )</th>
<th>( N = 64 )</th>
<th>( N = 81 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.86</td>
<td>0.98</td>
<td>1.19</td>
<td>0.78</td>
<td>0.98</td>
<td>1.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.82</td>
<td>0.71</td>
<td>0.8</td>
<td>0.72</td>
<td>0.73</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE I**

**Achieved fairness of LDA+ with \( N \) FTP and \( N \) LDA+ competing flows with RED and FIFO routers**

The results depicted in Tab. I indicate that \( F \) varies depending on the chosen simulation parameters between 0.7 and 1.2 for both RED and FIFO routers. These results suggest the TCP-friendliness of LDA+ under the simulated situation.

**B. Performance of LDA+ in the Presence of WWW Traffic**

of bursty traffic on the performance of LDA+ we used the topology depicted in Fig. 1 with \( (m = n = k = 27) \) and the bot-
tleneck bandwidth set to 10 Mb/s. The router used RED buffer management with the same parameters as in Sec. IV-A.

Tab. II presents the fairness results ($F$) with varying round trip delays ($\tau$) and different queuing delays. $F$ was determined as the rate of the LDA+ bandwidth share to the share consumed by the FTP connections. The presented results suggest that here as well LDA+ achieves acceptable friendliness values.

<table>
<thead>
<tr>
<th>$\tau$ (sec)</th>
<th>$s_{0}$</th>
<th>$s_{0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.65</td>
<td>0.86</td>
</tr>
<tr>
<td>0.4</td>
<td>0.91</td>
<td>1.13</td>
</tr>
</tbody>
</table>

**TABLE II**

**AVERAGE FAIRNESS WITH MIXED FTP, WWW AND LDA+ TRAFFIC**

For the case of short lived connections, however, those performance metrics can not be used. Short lived connections do not carry enough data in order to fully utilize a bandwidth share similar to that of a long lived TCP connection. With the WWW server model we are using as described in [14] the average connection carries about 20 packets. Hence, those short lived TCP connections will usually send their data in a few round trip times. To test the effects of introducing LDA+ flows to a network inhibited by short lived TCP connections we compare the bandwidth share those short lived connections can assume when competing with the LDA+ flows on the one hand and when competing with long lived TCP connections on the other. In this case, we expect that the bandwidth share of the short lived TCP connections should be in both cases rather similar.

For the simulation topology depicted in Fig. 1 with a RED router, round trip propagation delay ($\tau$) of 0.4 seconds, queuing delay of 0.1 seconds and link bandwidth of 10 Mb/s we ran two simulations with 27 FTP connections and 27 WWW servers in the first case and 27 LDA+ flow and 27 WWW servers in the second.

![Fig. 2](image1)

**Fig. 2.** Bandwidth sharing for the case of WWW traffic competing with FTP or LDA+ flows

Fig. 2 shows the bandwidth distribution for the case of WWW traffic competing with LDA+ flows on the one hand and with long lived TCP connections, i.e., FTP connections on the other. While the LDA+ flows receive a much higher bandwidth share than the WWW traffic, the LDA+ share is similar to that of the FTP share under the same conditions. This suggests that, adding LDA+ flows to the network does not affect the performance of the WWW traffic under the simulated situation any different than long lived TCP connections do.

V. SUMMARY AND FUTURE WORK

In this paper, we presented an algorithm for adapting the transmission rate of multimedia senders in a TCP-friendly manner using the RTP protocol for control purposes.

In addition to the here presented simulations we have conducted various other simulations and measurements testing the effects of using different values for the increase rate or when different senders start with different initial transmission rates or at different time points [15]. All of these simulations as well as a set of measurements on different links over the Internet suggest the efficiency of LDA+ in achieving high network utilization, avoiding losses and its TCP-friendly behavior towards competing TCP connections.

REFERENCES


