Adaptive Smooth Multicast Protocol for Multimedia Data Transmission

Christos Bouras, Apostolos Gkamas and Georgios Kioumourtzis

Abstract— **We introduce Adaptive Smooth Multicast Protocol (ASMP), for multimedia transmission over best-effort networks. The smoothness lays in the calculation and adaptation of the transmission rate, which is based on dynamic estimation of protocol parameters and dynamic adjustment of the "smoothness factor". ASMP key attributes are: a) adaptive scalability to large sets of receivers, b) TCP-friendly behavior, c) high bandwidth utilization, and finally d) smooth transmission rates, which are suitable for multimedia applications. We evaluate the performance of ASMP and investigate its behavior under various network conditions through extensive simulations, conducted with the network simulator software (ns2).**

Index Terms— **Multicast, congestion control, multimedia transmission, ns2, simulation.**

I. Introduction

Multicast transmission is a preferable solution of group communication applications such as multimedia applications, information dissemination services, software upgrade services etc. There are, though, many technical challenges and open issues that have to be addressed in the area of multicast transport protocols. RFC 2357 [1] describes the criteria that have to be taken into account when evaluating such protocols and concludes that it is unlikely that a single solution can be commonly accepted by all applications in today's Internet world. According to RFC 2357, the behavior of any proposed multicast transport protocol should be analyzed with respect to the following properties:

- Scalability: How scalable is the protocol to the number of senders or receivers in a group, to the number of groups, and wide dispersion of group members. What are the mechanisms that limit scalability?
- Congestion control: How the protocol addresses Internet congestion? How friendly is to TCP traffic?
- Error recovery: What are the mechanisms for error recovery?
- Security: How the protocol addresses a number of security and privacy concerns?

Multimedia applications, however, pose their own constraints and Quality of Service (QoS) requirements that in many cases increase the existing problem complexity concerning multicast transmission.

Research work, over the last few years, has offered new proposals in the field of multicast transport protocols. The proposed solutions have mainly concentrated to satisfy the congestion control, the TCP-friendliness, and the scalability related criteria. Recent work can be broadly classified into three main categories:

- Single Layer Design: The sender transmits a single layer and the transmission rate is defined by the lowest receiver, which is the one with the lowest bandwidth capacity.
- Layered transmission: Multimedia data is transmitted by a number of different streams and each individual receiver joins the multicast stream that is closer to its bandwidth capabilities.
- Replicated transmission: Under this approach the receivers form different multicast groups and each receiver joins the group that is closer to its bandwidth capabilities.

In this work we will concentrate on the single multicast transmission, as we believe it is the basis for the other aforementioned categories.

We present Adaptive Smooth Multicast Protocol (ASMP), a new single-rate multicast transport protocol for multimedia applications. The key attributes of our proposal are: a) adaptive scalability to large sets of receivers, b) TCP-friendly behavior, c) high bandwidth utilization, and finally d) smooth transmission rates, which are suitable for multimedia applications. In ASMP, each receiver calculates a TCP-friendly bandwidth share based on the TCP analytical model presented in [2]. The smooth behavior is naturally well suited to multimedia applications as high oscillations of the sending rates may create distortions of Audio-Video (AV) encoders and decoders. ASMP runs on top of the RTP/RTCP protocols [3] and uses the feedback sender and receiver reports for the dissemination of network related information between the sender(s) and the receivers. It is worth mentioning that ASMP does not require any additional support from the routers or the underlying IP-multicast protocols. This property allows easy deployment over unmanaged networks, like the Internet

Dr. Christos Bouras is Associate Professor in Computer Engineering and Informatics Department in University of Patras in Greece and Scientific Coordinator of Research Unit 6 in Research Academic Computer Technology Institute in Patras, Greece (corresponding author: Research Academic Computer Technology Institute, N. Kazantzaki Str., University of Patras, 26500 Rion, Greece, Tel: +30 2610 960375, Fax: +30 2610 960358, email: bouras@cti.gr).

Dr. Apostolos Gkamas is research engineer in Research Unit 6 of Research Academic Computer Technology Institute in Patras, Greece and visiting Lecturer in Department of Telecommunications Science and Technology of Peloponnesus University in Greece. (Research Academic Computer Technology Institute, N. Kazantzaki Str., University of Patras, 26500 Rion, Greece, Tel: +30 2610 960465, Fax: +30 2610 960358, email: gkamas@cti.gr)

Georgios Kioumourtzis is PhD candidate in Computer Engineering and Informatics Department in University of Patras in Greece (Presenter if paper accepted: Computer Engineering and Informatics Department, University of Patras, 26500 Rion, Patras, Greece, email: gkioumou@ceid.upatras.gr)

(where end to end QoS is not avalaible) which is the focus of our research.

The rest of this paper is organized as follows: Next section describes our motivation. We specify ASMP and describe its internal functions in section 3. Results from extensive ns2 [4] simulations are presented in Section 4. We conclude our paper in section 5.

II. Motivation

Significant research work and promising proposals have been submitted over the last few years in the field of single multicast layer transmission. TFMCC [5], extends the basic mechanisms of TFRC [6] to support single layer multicast congestion control. The most important attribute of TFMCC is the suppression of the feedback receiver reports. TFMCC is using the receiver with the lowest receiving capacity to act as the representative of the multicast group. PGMCC [7] is built on a different approach and uses a window-based TCP controller based on positive ACKs, between the sender and the group representative. TBRCA [8] targets at maximizing the overall amount of multimedia data to the whole set of receivers. With the use of a bandwidth rate control algorithm it dynamically controls the output rate of the video coder. LDA+ [9] employs a TCP equation based congestion control for measuring the TCP friendly bandwidth share in the event of packet losses. LDA+ uses the RTP protocol for collecting loss and delay statistics from the receivers.

Our motivation is to design an adaptive multicast protocol that can meet both the evaluation criteria posed by IETF in RFC 2357 and at the same time the QoS requirements and application constraints posed by multimedia applications. Additionally, we try to minimize the encoder/decoder distortion by avoiding rapid changes in the transmission rates of multimedia data. ASMP is intended to serve only multimedia applications as it has already been assessed by IETF that different applications have widely different requirements for congestion control and error recovery mechanisms. Our main concept is to exploit the functionality of an existing and widely used protocol in order to obtain the necessary network metrics for an adaptive and TCP-friendly behavior. We choose in our implementation RTP and its associate RTCP control protocol, which is the de facto standard protocol for multimedia transmission. RTP employs feedback suppression algorithms that increase and ensure scalability.

III. Protocol Description

ASMP is a single rate multicast protocol, which takes advantage of the RTCP feedback Sender (SR) and Receiver (RR) reports. The innovation in this work is the smooth transmission rate calculation, which is performed by the receivers and is based on both RTCP feedback reports and network statistics. Our main objective is to adjust the transmission rates in such way that oscillations are reduced and follow a smooth fashion. ASMP uses network statistical information based on network Congestion Indicators (CI) in order to tune the "smoothness factor". The adaptive algorithm adjusts the behavior of the protocol, making it less or more aggressive to upcoming network changes. Another important attribute is the long term TCP-friendliness, meaning that the multimedia stream consumes no more bandwidth than a TCP connection, which is traversing the same path with the multimedia stream. Moreover, with the use of the feedback RTCP reports we provide better scalability, as the amount of these feedback reports are controlled by the RTCP protocol and they cannot exceed a specified threshold, as percentage of the total available bandwidth [3]. Lastly, without disseminating any additional feedback reports than those of the RTCP Sender and receiver reports, we increase bandwidth utilization for user data. The only visible drawback in this approach is the long time intervals between two consecutive RTCP feedback reports. As a result, the sender does not have fast reactions when network conditions change very rapidly. However, our main objective under the RTP-based adaptation scheme is to adjust the sender's transmission rate to the average available bandwidth and prevent high oscillations of the transmission rate.

A high-level overview of the ASMP features is as follows:

- The receiver measures the loss event rate and the jitter delay based on the RTP packet sequence numbers and timestamps.
- The receiver measures the RTT to the sender based on the receiver's one-way time measurements and the sender RTCP feedback reports.
- The receiver measures the TCP friendly bandwidth share with the use of the analytical model of TCP.
- The receiver assesses the Congestion Indicators (CI) in order to adjust the "smoothness factor" and calculates a new smooth TCP-friendly transmission rate.
- The receiver sends this TCP-friendly transmission rate to the sender using the extension mechanism of RTP/RTCP.
- The sender adapts its transmission rate based on the RTCP feedback reports sent by all receivers that join the session.

More on the RTP/RTCP extensions can be found in [10]. In the following paragraphs we include a detailed description of the above functions.

A. *Measuring the Loss Event Rate*

The method for measuring the loss event rate is crucial for the TCP-friendly rate estimation. The receiver measures packet losses during an RTT interval, based on the functionality provided by the RTP protocol. The sequence numbers in the RTP packet header provide a straightforward way for the discovery of lost packets. In order to prevent a single spurious packet loss from having an excessive effect on the packet loss estimation, the receivers smooth the values of packet loss rate by using the following filter that computes the weighted average of the m most recent loss rate values l_i^m. The following filter has been presented and evaluated in [11], and provides a good estimation of the packet loss rate:

$$l_i^m = \frac{\sum_{i=0}^{m-1} w_i l_i}{\sum_{i=0}^{m-i} w_i} \quad (1)$$

where, l_i^m is the smooth value of packet loss rate for receiver i. The weights w_i are chosen so that very recent packet loss rates receive the same high weights, while the weights gradually decrease to 0 for older packet loss rate values. We use $m = 8$ and the following values for the weights:

$$w_i = \{1,1,1,1,0.8,0.6,0.4,0.2\} \quad (2)$$

B. *Measuring Jitter Delay*

Our implementation for delay jitter calculations is based on the algorithm defined in RFC 3550.

C. *Measuring the Round Trip Time (RTT)*

When a receiver i receives a RTP packet from the sender, it uses the algorithm described below in order to estimate the RTT between the sender and the receiver. Assuming that the sender and the receiver have synchronized clocks, the receiver can use the timestamp of the RTP packet ($T_{timestamp}$) and the local time when it receives that packet ($T_{receiver}$) to estimate the one-way delay, in the path between the sender and the receiver (T_{oneway}):

$$T_{oneway} = T_{receiver} - T_{timestamp} \quad (3)$$

If this path were symmetric and had the same delay in both directions, the RTT between the sender and the receiver would be twice T_{oneway}:

$$t_{RTT} = 2T_{oneway} \quad (4)$$

However, until now we have made two assumptions:

- The sender and the receiver have synchronized clocks.
- The path between the sender and the wired receiver is symmetric.

The above assumptions are not true for the Internet. Therefore, the receivers have to take the above assumptions into account in order to perform accurate RTT estimations (t_{RTT}). For this reason, we use a parameter α and we can write the equation (4) as:

$$t_{RTT} = (1+a)T_{oneway} \quad (5)$$

The parameter α is used to smooth the estimation of the RTT due to the potential unsynchronized clocks and the asymmetry of the path between the sender and the receiver.

To estimate the value of parameter α, the receivers need an effective estimation of the RTT, which can be acquired with the use of the RTCP reports. Thus, the RTCP receiver report contains two additional fields; the t_{LSR} (the timestamp of the most recent RTCP sender report) and the t_{DLSR} (the delay between the reception of the last sender report and the transmission of the receiver report). As a result the sender can make an effective RTT measurement for the path between the sender and a receiver by using the following equation. (A is the time when the sender receives the receiver report from the given receiver):

$$t_{RTT} = A - t_{LSR} - t_{DLSR} \quad (6)$$

The sender estimates an effective RTT measurement for a receiver i, every time it receives a receiver report from that receiver. It then includes this effective RTT measurement (with the id of the receiver) in the next RTCP sender report.

When a receiver receives the effective RTT measurement from the sender, it estimates an appropriate value for the parameter a by using the following equation:

$$a = \frac{t_{RTT}}{T_{oneway}} - 1 \quad (7)$$

Furthermore, in order to avoid solely phenomenon of an instant RTT high value, which will affect the RTT estimations, we use an exponentially weighted average value.

$$t_{RTT} = t_{RTT}^{inst} \cdot \beta + (1-\beta) \cdot t_{RTT} \quad (8)$$

where t_{RTT}^{inst} is the instantaneous RTT measurement made by the receiver and β a predefined value.. A high value of β provides more gravity to the instantaneous RTT measurement and results in more accurate RTT measurements. The trade-off of an instant and accurate RTT measurement is translated into higher oscillations of the calculated TCP-friendly rate. These oscillations, however, are not preferable by multimedia applications. Our performance evaluation shows that the selection of $\beta = 0.5$ offers a reasonable compromise between smooth transmission rate and responsiveness to network changes.

D. *TCP-friendly Bandwidth Share Estimations*

The receiver emulates the behavior of a TCP agent and as such when packet losses occur, it estimates a TCP friendly bandwidth share r_{tcp}^i every RTCP report interval, with the use of the following analytical model presented in [2].

$$r_{tcp}^i = \frac{P}{t_{RTT}\sqrt{\frac{2Dl}{3}} + t_{out}\min(1,3\sqrt{\frac{3Dl}{8}})l(1+32l^2)} \quad (9)$$

Where, r_{tcp}^i is the receiver's i estimation (in bytes/sec), P is packet size in bytes, l is the packet loss rate, t_{out} is the TCP retransmission timeout, t_{RTT} is the Round Trip Time (RTT) of the TCP connection and D is the number of acknowledged TCP packets by each acknowledgment. In our implementation we assume that $D = 1$ (each acknowledgment packet acknowledges one TCP packet) and $t_{out} = 4t_{RTT}$ (the TCP retransmission timeout is set to be four time the RTT). If the receiver has not experienced any packet losses since the previous RTCP report, the r_{tcp}^{i} must not be increased more than one P/RTT. For this reason the receiver calculates the new r_{tcp}^{i} value from the following equation (in bytes/sec):

$$r_{tcp}^{i} = r_{tcp}^{i} + \frac{1}{t_{RTT}} P \tag{10}$$

In addition, r_{tcp}^{i} cannot exceed the bottleneck bandwidth in the link between the sender and this receiver. We have implemented a Receiver Based Packet Pair (RBPP) algorithm ([12], [13]), which uses the RTP packet timestamps in order to measure the bottleneck bandwidth. With this algorithm the sender periodically transmits RTP packets in bursts. The receiver measures the time gaps between the arrivals of the two packets and calculates the bottleneck bandwidth (in bytes/sec) as follows:

$$B = \frac{P}{t_2 - t_1} \tag{11}$$

where, P is the packet size in bytes, t_2 and t_1 are the arrival times of the first and second packets, respectively. Next the receiver uses an additional filter that provides smoother transmission rate estimation.

$$r_{tcp}^{i} = r_{tcp}^{inst} \cdot \gamma + (1-\gamma) \cdot r_{tcp}^{i} \tag{12}$$

where, r_{tcp}^{inst} is the latest estimation of the transmission rate measured by the receiver, and γ a predefined value between 0 and 1.

$$0 \le \gamma \le 1 \tag{13}$$

The value γ should be carefully chosen as low values make the algorithm more insensitive to changing network conditions. For $\gamma = 1$, r_{tcp}^{i} is assigned the instantaneous measured value r_{tcp}^{inst}. Clearly, there is a trade-off between responsiveness to rapid network changes and smooth oscillations of the transmission rate. Our main objective is to prevent fast oscillations of the transmission rate. In this way we are able to better adapt and harmonize the transmission rate with the multimedia application constraints. We will elaborate in the next section, the behavior of the protocol under different γ values through simulation results.

E. Congestion Indicators (CI)

We have mentioned that the value of filter γ affects ASMP behavior in terms of a) responsiveness to rapid network changes, and b) smoothness of the estimated transmission rate. Therefore, we should adapt γ values in respect with the network congestion. Unfortunately, only packet loss ratio or only RTT values cannot provide a clear picture of the network congestion level [14]. Thus, we have implemented an early warning congestion algorithm, in which the receiver can detect upcoming network congestion based on statistical data. Our RTP-based implementation provides the raw data in the RTCP SR and RR reports, and it is up to the protocol designer on how to exploit and use this data. In our implementation we define the network status based on delay jitter measurements. We accept that the network can be on one of the following conditions:

- Condition CONGESTED: In this condition jitter delay has high values. Therefore, parameter γ should have a high value in order to respond rapidly to upcoming congestion.
- Condition LOADED: When the network is in this condition the transmission quality is good. Jitter delay is within predefined values that are affordable for multimedia applications. Parameter γ should have a low value to keep almost the same smooth transmission rate.
- Condition UNLOADED: In this condition jitter delay has minimum values. Parameter γ should have middle values to allow smooth increase of the transmission rate.

The receiver assesses the network condition every time it receives a RTP packet. The receiver compares the *z-score* of the new jitter value against predefined values. The *z-score*, which stands for the *standardized value* is defined as:

$$z = \frac{(Y - \bar{Y})}{s} \tag{14}$$

where, Y is the new estimated jitter value, $\bar{Y}$ is the mean of all previous jitter values, and s is the standard deviation. The standard deviation is defined as:

$$s = \sqrt{\frac{\Sigma(Y - \bar{Y})^2}{n-1}} \tag{15}$$

where, n is the number of estimated jitter samples.

In other words, the receiver estimates the “distance” of the new jitter value from the mean. If this “distance” is getting higher over time the receiver assesses that this is a congestion indication. If the distance is within an acceptable range the network can be viewed as *LOADED;* values below the mean indicate an *UNLOADED* network. Therefore, with the use of the above statistical data and a decision making algorithm the receiver defines parameter γ values as follows:

$$\begin{aligned} &if\,(-\delta \cdot s < z\_score < \delta \cdot s) \rightarrow LOADED \\ &esle\ if\,(z\_score <= -\delta \cdot s) \rightarrow UNLOADED \\ &esle\ CONGESTED \end{aligned} \tag{16}$$

where, δ defines the distance from the mean. We run several simulations with different values for parameter δ, based on the network topology of figure 3. We present in figure 1 how the different values of parameter δ affect ASMP’s behavior. We observe that δ values that are higher or lower than 1 produce sparks in the *z-scores*. These sparks indicate high congestion as jitter delay values seem to be 3 and 4 standard deviations away from the mean (figure 2). Therefore the meaning for us is that δ values, which are higher or lower than 1, make ASMP more responsive or insensitive to changes of jitter delay values. This is something that we need to control, so that ASMP is as responsive as required to jitter delay changes. Therefore, we define $\delta = 0.75$ as we observed that this value provides ASMP with the level of responsiveness we desire in order to obtain a smooth behavior.

We define the following values of parameter γ for the three different network conditions, based on simulation results:

$$LOADED \to \gamma = 0.1$$
$$UNLOADED \to \gamma = 0.3$$
$$CONGESTED \to \gamma = 0.5 \tag{17}$$

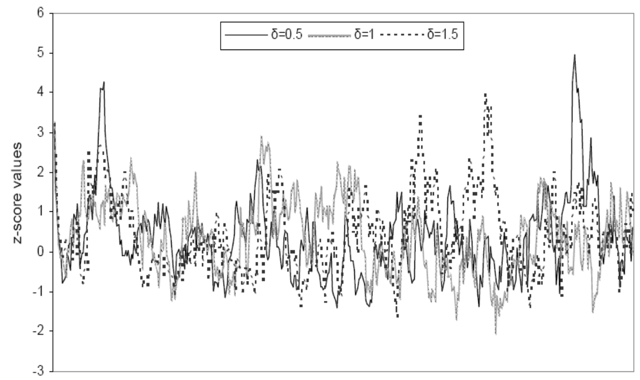

Fig. 1. Comparing z-scores for different δ values

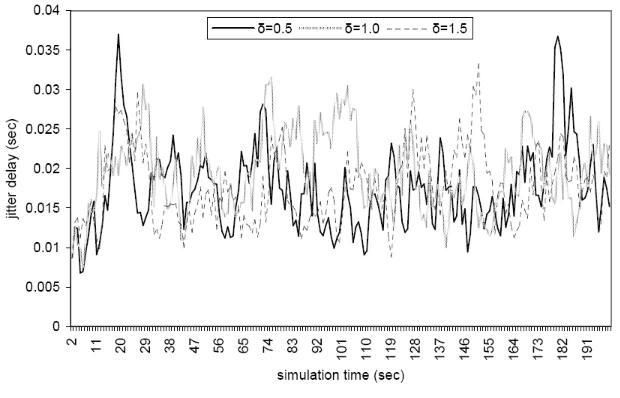

Fig. 2. Comparing jitter-delay for different δ values

F. Scalability

RFC 3550 recommends that control traffic in an active session should not be greater than 5% of the session bandwidth. This threshold, however, increases the time interval of the RTCP reports when the number of participants increases, as more participants share this 5% of the session bandwidth. The problem could have been solved if this threshold would increase in respect with the number of participants. Unfortunately, network resources are limited and we need to find other mechanisms that will absorb the side effects of large sessions with thousands of receivers. We implement an effective solution that was proposed and evaluated in [15].

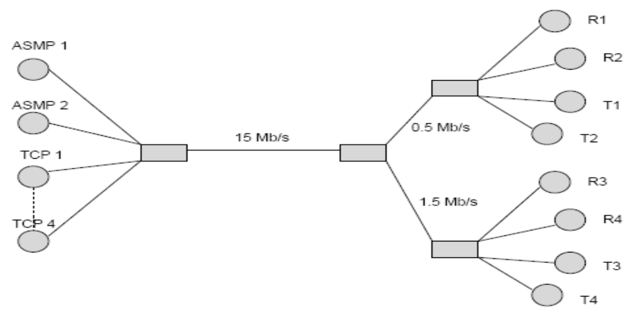

Fig. 3. TCP-fairness topology

IV. Performance Evaluation

We implement ASMP in ns2 to evaluate its performance under a controlled environment. We test the protocol under various simulation scenarios to investigate:

- The TCP-friendly behavior, when ASMP receivers share the same bottleneck link with multiple TCP connections.
- The responsiveness of the protocol to rapid changes of the network conditions.
- The behavior of the protocol when a "slow" receiver late joins the multicast session, and
- The responsiveness to changes of other competing data.

A. TCP-fairness

In this simulation we evaluate the fairness of ASMP towards competing TCP traffic. We use a simple bottleneck scenario in which two ASMP senders share multiple bottleneck links with four TCP Agents (Figure 3). R1, R2, R3 and R4 stand for the four ASMP receivers whereas T1, T2, T3 and T4 represent the TCP receivers. We set the initial rate of ASMP and TCP senders to 150 Kb/s. The bottleneck links should equally shared by ASMP and TCP flows, which means that the ASMP receiver must not consume more than 33.3% of the bottleneck bandwidth. Therefore, ASMP receivers in the low capacity link (0.5 Mb/s) must not consume more than 166.6 Kb/s, whereas in the higher link (1.5 Mb/s) they must not consume more than 499.5 Kb/s, in order to be TCP friendly. The rest of the available bandwidth must be shared by the two TCP connections and it is expected that each TCP connection will receive 166.6 Kb/s in the low capacity link (0.5 Mb/s) and 499.5 Kb/s in the higher link (1.5 Mb/s).

We use in our evaluation Random Early Drop (RED) queue in the routers to avoid synchronization in the routers' queues that will affect the simulation results.

Figure 4 depicts the achieved throughput during the simulation time. In the presented chart we have only two representative ASMP receivers from each group to show the achieved smooth rate reception contrary to TCP high oscillations.

We observe in figure 5 how the available bandwidth of the low capacity bottleneck link (0.5 Mb/s) is shared amongst the TCP and ASMP receivers. ASMP receivers have an average receiving capacity of 110.792 Kb/s, which is 66.50% of 166.6 Kb/s (this is the upper limit over which ASMP consumes more bandwidth that a competitive TCP connection). TCP receivers enjoy an even higher bandwidth share of 161.893 Kb/s

(97.17% of 166.6 Kb/s). In the case of "fast" receivers (figure 6), we obtain similar results. The ASMP average receiving rate is 374.582 Kb/s (75% of 500Kb/s), whereas the TCP receiver has higher throughput of 488.094 Kb/s (97.61%). Therefore, we conclude that ASMP is a TCP-friendly protocol that shares the available bandwidth with any competing TCP flow. ASMP is more TCP-friendly than it ought to be but this is an inherited property of the "smooth" algorithm and the high interval of the RTCP feedback reports. TCP on the other hand, with the embedded slow-start and feedback mechanisms reacts more rapidly to network changes. We explain the difference of ASMP throughput values between "low" and "fast" receivers as a result of the different feedback intervals. In our simulation scenario (figure 3) we keep the same number of ASMP receivers in the bottleneck links. Therefore, ASMP fast receivers have more available bandwidth for control traffic, which is interpreted into lower RTCP report intervals. These lower intervals make ASMP sender react faster to network changes.

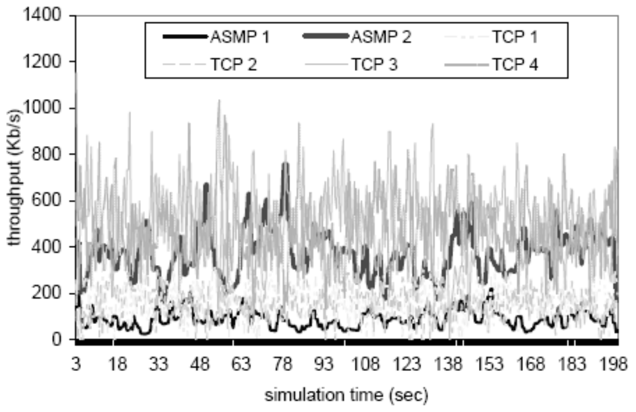

Fig. 4. Throughput of all flows

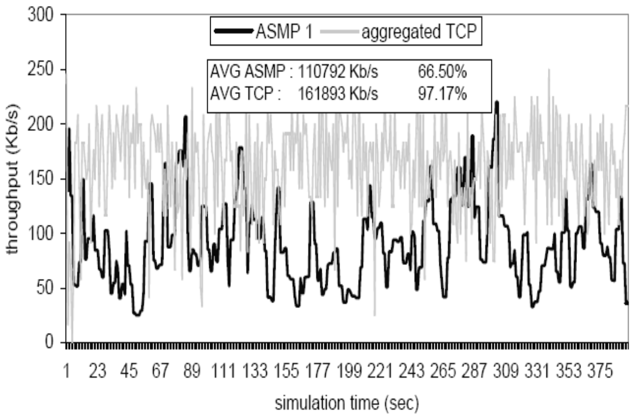

Fig. 5. Throughput of the 0.5 Mb/s bottleneck link

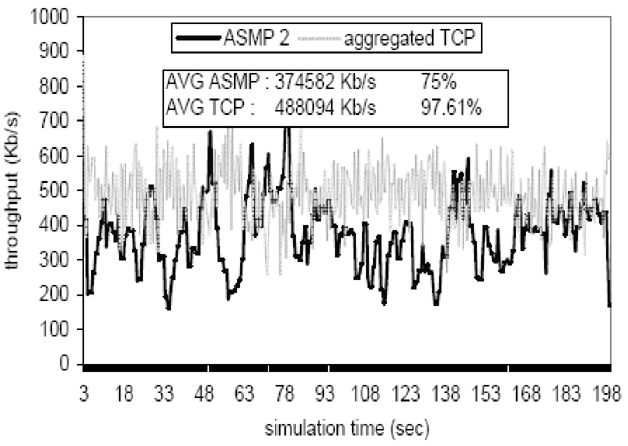

Fig. 6. Throughput of the 1.5 Mb/s bottleneck link

B. Responsiveness to Changes in the Loss Rate

In this simulation scenario we want to investigate how ASMP responds to changes in the loss rate and evaluate its performance. Loss rate variances affect the estimated TCP-friendly bandwidth share. We use a star topology with four links having loss rates of 0.05%, 0.08%, 0.12%, and 0.15%, respectively. At the beginning of the simulation we only have one receiver that joins the session and the other receivers join the session every 50-second intervals in the order of their loss rate. Receivers with lower loss rates in their links join first the session. After 200 seconds, receivers leave the session in reverse order; receivers with higher loss rates in their links leave first the session. TCP background traffic is transmitted in the four links along with ASMP traffic. Figure 7 depicts the simulation results. We observe how ASMP reacts when loss rates increase. In fact it takes the sender only few seconds to adjust the transmission rate to new conditions. ASMP transmission rates are similar to those of TCP, although they have are lower values.

For easier observation we present in figure 8 only one TCP flow of the link with loss rate equal to 0.12% in order to compare its transmission rates with those of ASMP in the same loss region. A striking difference is that ASMP transmission rates are always above zero (they are in fact in the scale of few KB/s), as a result of the smooth algorithms that are implemented in ASMP.

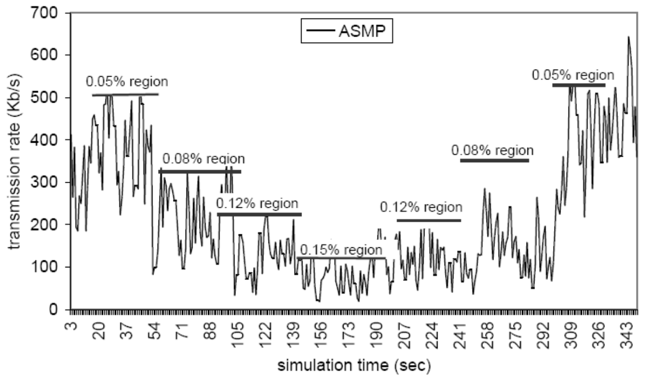

Fig. 7. ASMP transmission rates with respect to changes in the loss rate

The above experiment demonstrates that ASMP has very good responsiveness to changes of the network congestion level. In the following simulation we will observe how ASMP adjusts its transmission rate in the event of a join of a low receiver.

C. Late Join of Low-Rate Receiver

The case in multicast transmission is not meant by any means that all receivers have similar receiving capacity.

Therefore, the sender should adjust the transmission rate even though a low capacity receiver joins a session in which other receivers are connected with high capacity links. ASMP should be able to adjust the transmission rate under these conditions. This is the scenario that we will examine in this simulation.

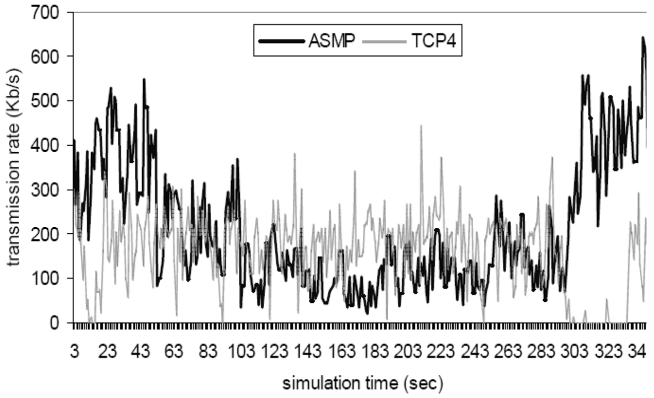

Fig. 8. ASMP and TCP transmission rates

Our multicast session consists of five ASMP receivers and four TCP flows. All receivers (ASMP and TCP) are connected with the senders via a 1.5 Mb/s link. In the beginning of the simulation the ASMP sender transmits at a rate of 500 Kb/s and only four out of five ASMP receivers, with similar bandwidth capacity, join the session. At the 50th simulation second the fifth low capacity ASMP receiver, which is connected to the network with a 300Kb/s link, joins the session. We observe that ASMP can handle this situation by adjusting the transmission rate in respect with the low receiver's bandwidth capacity (figure 9). At the 100th simulation second the low capacity receiver leaves the session and the ASMP sender rapidly adjusts its transmission rate.

We conclude in this simulation scenario that not only ASMP keeps a smooth transmission rate but also reacts rapidly to network topology changes.

D. Responsiveness to Dynamics of Competing Traffic

We test ASMP responsiveness to changes of competing traffic. We run a simple bottleneck simulation scenario in which the bottleneck link (1Mb/s) is shared by one ASMP stream and two TCP flows. The first TCP flow (TCP1) is transmitted in the beginning of the simulation at the same time

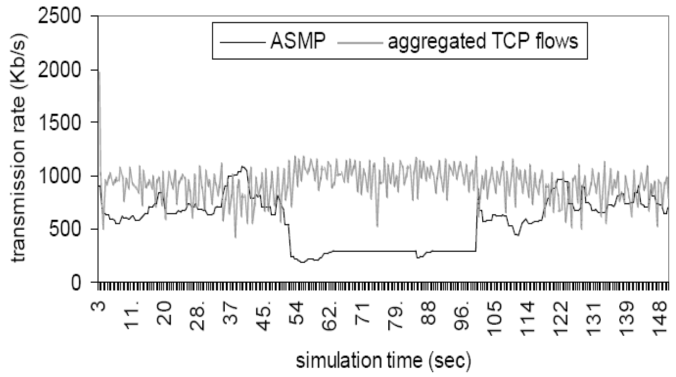

Fig. 9. ASMP transmission rates in respect with late join of low-rate receiver

with the ASMP stream. Both, ASMP and TCP senders are transmitting at 300 Kb/s. The second TCP flow (TCP2) starts transmitting at the 30th simulation time with a rate at 300 Kb/s. Up to this point we have three flows in the bottleneck link that require 900 Kb/s. We observe in the simulation results (figure 10) that the link is almost equally shared by the three flows, as the congestion is rather mild. At the 80th simulation second we add an additional flow in the bottleneck link. A Constant Bit Rate (CBR) application starts transmitting at 300 Kb/s, which now causes high congestion. However, it takes ASMP only few seconds to adjust its transmission rate to the new conditions, and the transmission rates never drop to zero during the simulation lifetime. In the worst-case, ASMP transmission rate is around 100 Kb/s. TCP2 stops its transmission at the 120th simulation second. We observe that ASMP increases its transmission rate from that time. The CBR application stops its transmission at the 170th simulation second. ASMP has again a quick reaction and increases the transmission rate as it encounters better conditions in the bottleneck link.

With this experiment we conclude that the impact of the smooth algorithms in ASMP is not big as the protocol has the ability to early detect upcoming congestion and adjust its transmission rate accordingly.

V. Conclusions-Future Work

We presented in this work ASMP, which is a new approach for multicast transmission of multimedia data. ASMP does not restrict the calculation of the TCP-friendly share with only the use of RTT and loss ratio values. ASMP filters this calculated rate in a dynamic way based on statistical data of jitter delay measurements. ASMP feedback control functions are based on an existed and well-accepted protocol. The RTCP sender and receiver reports eliminate the need for additional feedback mechanism. Simulation results show that ASMP performs well under harsh network conditions with high loss rate links and dynamic changes of other competing traffic. Scalability is ensured by both the internal control functions of the RTCP protocol and additional feedback suppression mechanisms.

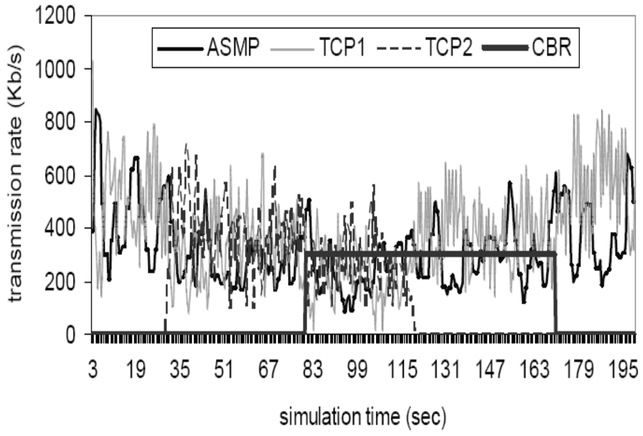

Fig. 10. ASMP responsiveness to dynamics of competing traffic

Our future work includes the comparison of ASMP with other single-rate multicast protocols under the same network conditions. We will also make the needed modifications (if any) to apply ASMP in wireless environments in which jitter delay measurements exploit better the network congestion level than packet loss events.

Acknowledgment

We thank the anonymous SPECTS 2008 reviewers for their helpful comments.

References

[1] RFC 2357: A. Mankin, A. Romanow, S. Bradner, V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", June 1998.

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

[3] RFC 3550, "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, July 2003.

[4] http://www.isi.edu/nsnam/ns/

[5] J. Widmer and M. Handley, "Extending equation-based congestion control to multicast applications," *in Proc. of ACM SIGCOMM '01*, 2001.

[6] RFC 3448, M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC)", *Network Working Group*, January 2003.

[7] L. Rizzo, "pgmcc: A TCP-friendly single-rate multicast congestion control scheme," *in Proc. of ACM SIGCOMM '00*, 2000.

[8] Smith, H., Mutka, M., Rover, D. A Feedback based Rate Control Algorithm for Multicast Transmitted Video Conferencing, *Accepted for publication in the Journal of High Speed Networks.*

[9] Sisalem D., Wolisz A., "LDA + TCP - Friendly Adaptation: A Measurement and Comparison Study," *in the 10th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'2000),* June 25 - 28, 2000, Chapel Hill, NC, USA.

[10] C. Bouras, A. Gkamas, G. Kioumourtzis, "Extending the Functionality of RTP/RTCP Implementation in Network Simulator (ns – 2)" *First International Conference on Simulation Tools and Techniques for Communications, Networks, and Systems*, Marseille, France, 3 - 7 March 2008.

[11] L. Vicisiano, L. Rizzo, J. Crowcroft, "TCP - like congestion control for layered multicast data transfer", *in IEEE INFOCOM*, March 1998, pp. 996 - 1003.

[12] Srinivasan Keshav. "A control-theoretic approach to flow control". *In Proceedings of SIGCOMM,* 1991.

[13] Vern Paxson, "Measurements and Analysis of End-to-End Internet Dynamics", *PhD thesis, University of California,* Berkeley, April 1997.

[14] C. Bouras, A. Gkamas," Streaming multimedia data with adaptive QoS characteristics", *5th International Conference on Protocols for Multimedia Systems-PROMS 2000*, Cracow, Poland, 22 - 25 October 2000, pp. 129 – 139.

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