INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. (2009)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac.1059

Adaptive smooth multicast protocol for multimedia transmission: Implementation details and performance evaluation

Christos Bouras[1,*,†], Apostolos Gkamas[1] and Georgios Kioumourtzis[2]

[1]*Research Academic Computer Technology Institute and University of Patras, N. Kazantzaki Str., University of Patras, 26504 Rion, Greece*
[2]*Department of Computer Engineering and Informatics, University of Patras, 26500 Rion, Patras, Greece*

SUMMARY

In this paper we propose a new single-rate multicast congestion control scheme named 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 estimations of protocols' parameters and dynamic adjustment of the 'smoothness factor', as well. ASMP key attributes are: (a) TCP-friendly behavior, (b) adaptive scalability to large sets of receivers, (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). Simulation results show that ASMP can be regarded as a serious competitor of TFMCC and PGMCC. In many cases, ASMP outperforms TFMCC in terms of TCP-friendliness and smooth transmission rates, while PGMCC presents lower scalability than ASMP. We have implemented ASMP on top of RTP/RTCP protocols in ns2 by adding all the RTP/RTCP protocol's attributes that are defined in RFC 3550 and related to quality of service metrics. Copyright © 2009 John Wiley & Sons, Ltd.

Received 14 February 2009; Revised 3 May 2009; Accepted 23 July 2009

KEY WORDS: multimedia transmission; TCP-friendly; multicast; RTP/RTCP protocol; network simulator (ns-2)

1. INTRODUCTION

Multicast transmission is a preferable solution of group communication applications such as multimedia applications, information dissemination services, software upgrade services, etc. There are, however, many technical challenges and open issues that have to be addressed when designing transport protocols for multicast transmission.

*Correspondence to: Christos Bouras, Research Academic Computer Technology Institute and University of Patras, N. Kazantzaki Str., University of Patras, 26504 Rion, Greece.
†E-mail: bouras@cti.gr

RFC 2357 [1] describes the criteria that have to be taken into account for the evaluation of 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?

The aforementioned problem complexity increases when one tries to accommodate any multicast transport protocol for multimedia data transmission. The main reason is that multimedia applications pose their own constraints and Quality of Service (QoS) requirements, as a direct result of its nature. These applications are different from TCP-based applications and characterized mainly by the following three properties: (a) the demand for high data transmission rate (bandwidth-consuming applications), (b) the sensitiveness to packet delays (latency and jitter), and (c) the tolerance to packet losses (packet loss tolerant applications). Moreover, congestion control and TCP-friendliness, for instance, pose additional design requirements as shark teeth-like transmission rates may be too difficult to adjust by Audio–Video (AV) encoders and decoders. For a multimedia application, smooth and steady transmission rates with low inter-arrival jitter delay are more important attributes than guaranteed and on order delivery of data packets. Therefore, our motivation is to design a 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. This protocol is intended to serve only multimedia data transmission, as it has already 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 RTP [2] and its associate RTCP control protocol, which is the *de facto* standard form multimedia transmission in networks today. RTP employs feedback suppression algorithms that increase scalability.

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) TCP-friendly behavior, (b) adaptive scalability to large sets of receivers, (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 [3]. 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. The use of large buffers in order to overcome high oscillations of the sending rate is not always acceptable especially for multimedia applications with high interactivity like videoconference. ASMP runs on top of the RTP/RTCP protocols 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 an easy deployment over unmanaged networks like the Internet.

The remainder of this paper is organized as follows: next section discusses related work. We specify ASMP and describe its internal functions in Section 3. Results from extensive ns2 [4] simulations are presented in Section 4. We compare ASMP against TFMCC and PGMCC in Section 5. We conclude our paper in Section 6. Finally, we discuss the future work in Section 7.

2. RELATED WORK

Research work in the area of multicast transmission can be classified into two main categories in terms of the number of transmitted layers in the multicast session:

- *Single Layer Design*: In this category the sender transmits a single layer and the transmission rate is defined by the receiver with the lowest bandwidth capacity.
- *Multi-Layer Design*: Under this approach multimedia data are transmitted by a number of different streams and each individual receiver joins the multicast stream(s) that is closer to its bandwidth capabilities.

We will briefly discuss the related work of each area in the subsections below.

2.1. *Single-rate schemes*

There have been promising approaches in the single-rate area, over the last few years. An early work, TFMCC [5, 6], extends the basic mechanisms of TFRC [7] to support single stream multicast congestion control. The most important attribute of TFMCC is the suppression of feedback reports. TFMCC uses the receiver with the lowest receiving capacity acting as the representative of the multicast group. The sender adjusts the transmission rate based on feedback reports from the group representative. PGMCC [8] is a window-based TCP scheme, which is based on positive ACKs between the sender and the group representative (the acker). Only the acker is tasked to send positive ACKs to the sender, miming the 'classic' TCP receiver. All other receivers in the multicast session send NACKs whenever they discover packet losses. TBRCA [9] targets at maximizing the overall amount of multimedia data to the whole set of receivers and at the same time to serve receivers with low bandwidth connections. With the use of a bandwidth rate control algorithm, it dynamically controls the output rate of the video coder. LDA+ [10] is an additive increase and multiplicative decrease (AIMD) algorithm, in which the addition and reduction values are dynamically determined based on the current network conditions. To do so, LDA+ uses feedback from the receivers that are based on the RTP protocol. LDA+ employs the TCP analytical model in order to estimate a TCP-friendly bandwidth share in the event of packet losses. LDA+ does not implement any additional feedback suppression mechanism except for the one that is provided by the RTCP protocol. MDP-CC [11] also uses representatives for the adaptation of the transmission rate by the sender. The difference is that MDP-CC maintains a pool of representative candidates for the representative selection. ERMCC [12] implements a congestion control scheme that is based on a new metric named throughput rate at congestion. ERMCC can be implemented only at the sender and the receivers of the multicast group without any network support. The sender dynamically selects one of the slowest receivers as Congestion Representative (CR), and only considers their feedback reports for the adaption of the transmission rate.

The feedback suppression philosophy is similar to that of TFMCC, although it seems to offer higher scalability.

2.2. *Multi-rate schemes*

As we mentioned previously, multimedia data in multi-rate congestion control schemes are transmitted by a number of different multicast streams, which are named as 'layers'. Each individual receiver joins either the multicast group that is closer to its receiving capacity or receives multiple cumulative layers. Therefore, multi-rate schemes are well fitted to heterogeneous environments, in which they can succeed better bandwidth utilization.

Receiver-driven Layered Multicast (RLM) [13] is an early work in which receivers are able to join different multicast groups, with respect to the observed congestion in the network. The sender has no any active participation in the congestion control process except for the transmission of the different layers in separate multicast groups. Thus, receivers add an additional layer, which increase the quality of multimedia data, when they observe spare link capacity. Alternately, they drop the additional layer when they observe congestion. Join and leave requests are implemented with the use of IGMP [14]. In PLM [15], which stands for Packet pair receiver-driven Layered Multicast, receivers add and drop layers in terms of the available bandwidth in the bottleneck link. That was an improvement of RLM in order to mitigate the effects of falsely adding a higher layer that increases congestion. With the use of packet pairs, receivers normally would not add any layer that leads to higher receiving rates than the bottleneck link. The sender again in PLM has the same passive role, which is restricted to the transmission of the different layers. RLC [16] follows the same philosophy with previous work but seems to improve TCP-friendliness. FLID-DL [17] tries to mitigate known drawbacks that are related to long IGMP leave latencies, which leave the network in a congested state. FLID-DL uses dynamic layers in a sense that the sender changes the transmission rate of each layer over time. Receivers in FLID-DL can decrease their transmission rates by not joining any additional layers. In order for receivers to maintain a given reception rate they must periodically join additional layers at a moderate pace.

In all the previous proposals the sender plays a passive role and the various layers have fixed transmission rates. Fine-grained layered multicast [18] uses a different approach in an effort to minimize the drawbacks of cumulative layering regarding the coarse-grained of congestion control. Fine-grained proposed a non-cumulative layer join in which receivers infer the level of congestion in the network, by using packet losses, and adjust their subscription level accordingly. STAIR (Simulate TCP's AIMD with Rate-based) [19] further minimizes the IGMP control traffic with the concept of 'stair layers'. The stair layer is the layer whose rate dynamically increases over time from a base rate of one packet per RTT up to a maximum rate, before dropping back to the base rate. STAIR simulates the behavior of a TCP unicast connection with the use of an AIMD congestion control algorithm. However, it introduces high oscillations of the transmission rates with the well-known saw tooth pattern.

Concluding our review to related work, we justify that the single layer category is simple to design and implement, and in general does not require intermediate network support. The only visible drawback is the underutilization of network recourses (bandwidth) because the sender's transmission rate depends on the receiver with the 'slowest' receiving capacity. However, single-rate transmission mechanisms cannot be only seen as independent schemes, but also as building blocks of multi-rate schemes. SMCC [20] and GMCC [21] are good examples of multi-rate schemes that are based on single-rate transmission schemes.

In this paper we focus on the field of single-rate design. The discussion of the limitations of single-rate multicast protocols versus multi-rate protocols is out of the scope of this paper.

3. ADAPTIVE SMOOTH MULTICAST PROTOCOL (ASMP) DESCRIPTION

ASMP consists of a single-rate multicast congestion control, which takes advantage of RTCP Sender (SR) and Receiver Reports (RR). The innovation in this work is the calculation of smooth transmission rate, which is performed by receivers and is based on RTCP reports. Our main objective is to adjust the sender's transmission rate in such way that oscillations are reduced, following a smooth fashion. 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 RTCP feedback 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 [2]. Without disseminating any additional feedback reports (ACKs or NACKs) than those of RTCP sender and receiver reports, we increase bandwidth utilization for user data. Lastly, we develop ASMP on top of RTP/RTCP protocol. This approach has several advantages because we move the complexity up to the application layer, leaving untouched the operating system and network elements, as well. RTP with TCP-friendly control [22] is also built on this approach. The only visible drawback that we can see is related to high time intervals between two consecutive RTCP feedback reports in a large multicast group. As a result, the sender cannot react quickly when network conditions change very rapidly. However, we increase responsiveness with the use of network statistical information. This information is based on what we call Congestion Indicators (CI), which provide the warnings of upcoming congestion so that the 'smoothness factor' is tuned to a proper value. The smoothness factor regulates the behavior of the congestion control mechanism, making it less or more aggressive, with respect to the level of congestion in the bottleneck link. In this way, we succeed to have a congestion control that is 'smooth', without suffering from high oscillations, and at the same time very responsive to network changes. We will observe in the simulation results how the proposed scheme reacts to different events that cause changes to network conditions.

A high-level overview of the functionality of ASMP is presented below:

- The receiver measures the loss event rate and the jitter delay based on RTP packet sequence numbers and timestamps.
- The receiver measures the RTT between itself and the sender based on receiver's one-way time measurements and the sender RTCP feedback reports.
- The receiver measures a TCP-friendly bandwidth share with the use of the analytical model of TCP in Equation (12).
- The receiver is notified by CI in order to adjust the 'smoothness factor', and calculates a new smooth TCP-friendly transmission rate.
- The receiver sends the calculated smooth TCP-friendly rate to the sender using the extension mechanisms of RTP/RTCP application specific part (APP) of the RTCP.
- The sender adjusts its transmission rate based on RTCP feedback receiver reports.

In the following paragraphs we include a detailed description of the above functions.

3.1. ASMP sender

In ASMP the sender is responsible for two major functions: (a) calculation of the RTT in the link between the sender and each receiver and (b) adaptation of the transmission rate, so that the slowest receiver is not overloaded. In order to perform the first function, ASMP uses a specific application part in the RTCP sender report in which it reports the measured RTT for receiver i, and the receiver's i identification number. The sender stores the last values of r^i_{tcp} from all the receivers (added by the receivers in the application part of the RTCP) with each incoming RTCP receiver report. The adaptation of the transmission rate is the result of comparison of the current reported TCP-friendly estimation of receiver i to the latest estimations sent by all receivers. If the current estimated value is less than all latest reported estimations, the sender will adjust its transmission rate according to:

$$r_{tx} = \min(r^1_{\text{tcp}}, \ldots, r^i_{\text{tcp}}) \tag{1}$$

where r_{tx} is the new transmission rate. Figure 1 shows the flow chart of sender's functions. We will discuss the details of the various functions of ASMP sender in the following paragraphs.

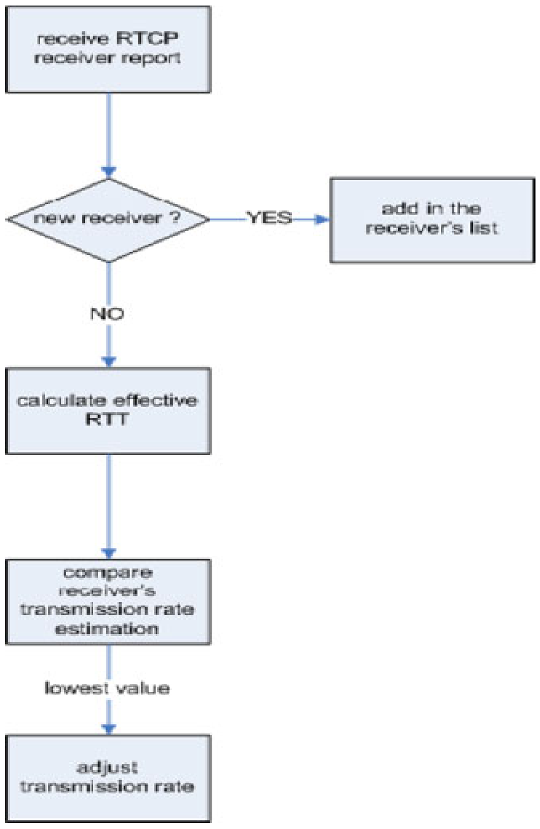
receive RTCP receiver report
new receiver ?
YES
add in the receiver's list
NO
calculate effective RTT
compare receiver's transmission rate estimation
lowest value
adjust transmission rate

Figure 1. Operations at ASMP sender.

3.2. ASMP receiver

The ASMP receiver is responsible for the following major functions: (a) RTT estimations, (b) adjustment of the 'smoothens' factor, (c) packet loss ratio calculations, and finally (d) TCP-friendly transmission rate calculations. The receiver emulates the behavior of a TCP agent with the use of the analytical model of TCP and estimates a TCP-friendly bandwidth share r^i_{tcp} every time it sends an RTCP receiver report. Figure 2 shows the flow chart of receiver's functions. We will also discuss the details of the various functions of the ASMP receiver in the following paragraphs.

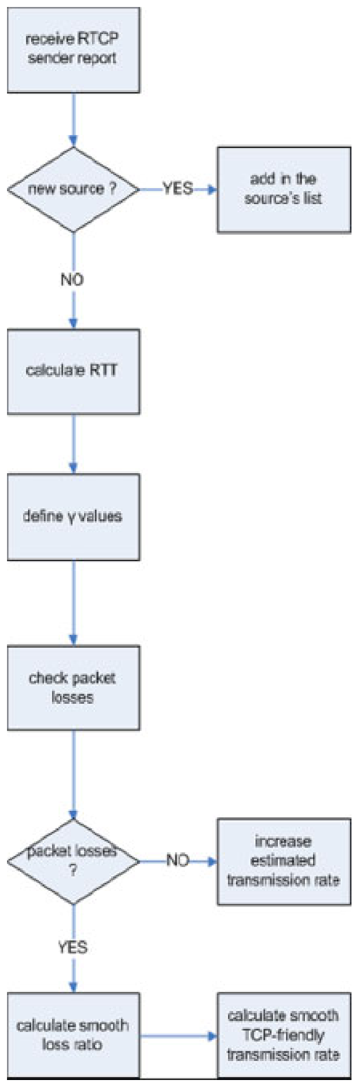
receive RTCP sender report
new source ?
YES
add in the source's list
NO
calculate RTT
define γ values
check packet losses
packet losses ?
NO
increase estimated transmission rate
YES
calculate smooth loss ratio
calculate smooth TCP-friendly transmission rate

Figure 2. Operations at ASMP receiver.

3.3. 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 RTCP report 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, 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:

$$l_i^m = \frac{\sum_{i=0}^{m-1} w_i l_i}{\sum_{i=0}^{m-i} w_i} \tag{2}$$

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. The value m determines the speed of ASMP in responding to network congestion. Large values of m would increase the ASMP's responding time when competing for network resources with TCP. In our implementation we use $m=8$ and the following values for the weights based on the recommendations given in [7]:

$$w_i = \{1, 1, 1, 1, 0.8, 0.6, 0.4, 0.2\} \tag{3}$$

3.4. Measuring jitter delay

Our implementation for jitter delay calculations is based on the algorithm defined in [2]. Shortly explaining, let S_i is the RTP timestamp of packet i, and R_i is the arriving time in RTP timestamp units of packet i, then for two sequentially packets i and j, delay D may be expressed as:

$$D = (R_j - R_i) - (S_j - S_i) \tag{4}$$

This delay variation should be calculated for each RTP packet. RFC 3550 suggests a filter function to avoid temporal fluctuation and thus the delay jitter is computed with the use of the following equation:

$$J_i = (\tfrac{15}{16}) J_{i-1} + (\tfrac{1}{16}) D \tag{5}$$

3.5. Measuring the round trip time (RTT)

When a receiver i receives an RTP packet from the sender, it uses the algorithm described below in order to estimate the RTT between itself and the sender. The receiver can use the timestamp of the RTP packet ($T_{\text{timestamp}}$) and the local time (T_{receiver}) to estimate the one-way delay (T_{oneway}) in the path between itself and the sender:

$$T_{\text{oneway}} = T_{\text{receiver}} - T_{\text{timestamp}} \tag{6}$$

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

$$t_{\text{RTT}} = 2T_{\text{oneway}} \tag{7}$$

The above assumptions are not true for the Internet. Therefore, 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 Equation (7) as:

$$t_{\mathrm{RTT}} = (1+a)T_{\mathrm{oneway}} \tag{8}$$

The parameter is used to smooth the estimated RTT due to the potential unsynchronized clocks and the asymmetry of the path between the sender and the wired receiver. To estimate the value of parameter a, receivers need an effective estimation of RTT, which can be acquired, with the use of RTCP reports: The RTCP receiver report contains two additional fields; the t_{LSR} (the timestamp of the most recent RTCP sender report from the sender) 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 perform an effective RTT measurement for the path between itself and a receiver by using the following equation (A is the time when the sender receives the receiver report from the given receiver):

$$t_{\mathrm{RTT}}^{r-i} = A - t_{\mathrm{LSR}} - t_{\mathrm{DLSR}} \tag{9}$$

The sender estimates an effective RTT measurement for 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:

$$a = \frac{t_{\mathrm{RTT}}^{r-i}}{T_{\mathrm{oneway}}} - 1 \tag{10}$$

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.

$$t_{\mathrm{RTT}} = t_{\mathrm{RTT}}^{\mathrm{inst}} \cdot \beta + (1-\beta) \cdot t_{\mathrm{RTT}} \tag{11}$$

where $t_{\mathrm{RTT}}^{\mathrm{inst}}$ 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. Therefore, we define this value of β for all of the simulations that are presented later in this paper.

3.6. 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} in every RTCP report interval, with the use of the following analytical model presented in [3]:

$$r_{\mathrm{tcp}}^{i} = \frac{P}{t_{\mathrm{RTT}}\sqrt{\dfrac{2Dl}{3}} + t_{\mathrm{out}} \min\left(1, 3\sqrt{\dfrac{3Dl}{8}}\right) l(1+32l^2)} \tag{12}$$

where r_{tcp}^{i} is the receiver's i estimation (in bytes/s), 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 $t_{\text{out}}=4t_{\text{RTT}}$ (the TCP retransmission timeout is set to be four time the RTT). In order to avoid abrupt changes of the transmission rate we define that when 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 we define the following equation for the receiver calculation about the new r_{tcp}^{i} value (in bytes/s):

$$r_{\text{tcp}}^{i}=r_{\text{tcp}}^{i}+\frac{1}{t_{\text{RTT}}}P \tag{13}$$

Next we define the following function that provides smoother transmission rate estimations:

$$r_{\text{tcp}}^{i}=r_{\text{tcp}}^{\text{inst}}\cdot\gamma+(1-\gamma)\cdot r_{\text{tcp}}^{i} \tag{14}$$

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

$$0\leqslant\gamma\leqslant 1 \tag{15}$$

The value γ should be carefully chosen as low values make the algorithm more insensitive to the changing network conditions. For $\gamma=1$, r_{tcp}^{i} is assigned the instantaneous measured value $r_{\text{tcp}}^{\text{inst}}$. In our previous work [23], we investigated the effects of different γ values on the ASMP performance. 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.

3.7. 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 with respect to the level of congestion in the network. Unfortunately, only the packet loss ratio or only the RTT values cannot provide a clear picture of the network congestion level [24] and when packet losses are observed the network is already congested. Thus, to avoid packet losses we need to implement an early warning congestion algorithm, in which—in the ideal case—the receiver can detect upcoming network congestion prior to any packet losses. The 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 these data to detect upcoming congestion. In our implementation we assess the network status based on cumulative jitter measurements [25]. We accept that the network can be on one of the following conditions:

- *Condition CONGESTED*: In this condition, cumulative jitter delay has increasing values over time as jitter delay is also increasing over time. Therefore, we need to adjust the sender's transmission rate in such way as to prevent congestion in the bottleneck link that will lead to packet losses. In order to do so we should give a low value to parameter γ, so that we can regulate the increase in the transmission rate and make it less than one packet per RTT.

Otherwise, if the sender continues to increase the transmission rate at a constant rate by one P/RTT it will soon cause congestion and packet losses.

- *Condition UNLOADED*: In this condition cumulative jitter has almost constant values. Parameter γ should have values close to one to allow a constant increase of the transmission rate that is close to one P/RTT.

An effective way to assess the level of congestion is to compare the long-running average of cumulative jitter against a *short running average*. When the short running average is bigger than the long running average the congestion level is increasing which means that we meet the condition CONGESTED. Conversely, when the short running average is smaller than the long running average we asses that the network is UNLOADED.

The cumulative average of the sequence of i values $CJ_1, \ldots, CJ_i$ up to the current time is defined as:

$$A_i = \frac{CJ_1 + \cdots + CJ_i}{i} \tag{16}$$

When a new value CJ_{i+1} becomes available, we update the cumulative average using the formula:

$$A_{i+1} = A_i + \frac{CJ_{i+1} - A_i}{i+1} \tag{17}$$

where the A_0 can be taken to be equal to zero. The above calculations present very low complexity every time a new data point is added. The *long running average* takes into account all the cumulative jitter values since the start of the transmission until the current time. The *short running average* is measured between two consecutive RTCP reports. The receiver assesses the congestion level every time it generates an RTCP feedback report with the following algorithm:

$$\begin{aligned} &\textit{if}\ (A_{short\_avg} \geqslant A_{long\_avg})\ \textit{then}\ \{ \\ &\quad \textit{smoothness\_factor} = \textit{CONGESTED} \\ &\}\ \textit{else smoothness\_factor} = \textit{UNLOADED} \end{aligned} \tag{18}$$

The RTCP timeout interval for the generation of a new RTCP report is dynamically defined by the RTCP protocol in accordance with the number of participants in a multicast session. Therefore, the sampling window is large in multicast groups with a large number of receivers and low in groups with a low number of receivers. In this way we can better correlate the congestion estimation algorithm with the number of participants in a multicast session as in groups with a large number of receivers the sender receives randomly a fairly large amount of RTCP feedback reports. For our simulations that are presented later in this paper we define the following values of parameter γ based on simulation results for the two different network conditions:

$$\begin{aligned} \textit{UNLOADED} &\rightarrow \gamma = 0.9 \\ \textit{CONGESTED} &\rightarrow \gamma = 0.2 \end{aligned} \tag{19}$$

3.8. Scalability

RFC 3550 recommends that the 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 with respect to the number of participants. Unfortunately, network resources are limited and we need to find other mechanisms that will mitigate the side effects of large sessions with thousands of receivers.

We implement an effective solution that was proposed and evaluated in [26]. Under this scheme, when a receiver encounters a high retransmission interval, which is defined as greater than T_{suspent}, (which means that the number of participants has increase too) it uses a truncated exponentially distributed retransmission timer in the interval $[0, T_{\text{rand}}]$, with density of:

$$\begin{aligned} T_{\text{wait}} &= \frac{1}{e^{\lambda}} \cdot \frac{\lambda}{T_{\text{random}}} \cdot e^{(\lambda / T_{\text{random}}) \cdot z} && \text{if } 0 \leqslant z \leqslant T_{\text{random}} \\ T_{\text{wait}} &= 0 && \text{otherwise} \end{aligned} \tag{20}$$

Each receiver schedules the RTCP retransmission timeout to be T_{wait}. If the receiver intercepts a receiver report that was just sent by another member of the multicast session, with a TCP-friendly share similar to its estimation, it suspends the RR transmission (we consider that two TCP-friendly bandwidth shares are similar when they differ only up to 2%). It is shown in [26] that with the proper selection of parameters the number of feedback messages is dramatically reduced, while keeping the feedback evaluation mechanism up-to-date. Therefore, the sender is still able to collect the necessary feedback information from a representing part of the receivers.

4. PERFORMANCE EVALUATION

In this work we built ASMP on top of the RTP in ns2 to evaluate its performance under a controlled environment. The legacy RTP implementation in ns2 is very generic and provides only the methods for the creation of sessions between the sender and the receivers. The RTCP sender and receiver feedback reports do not provide any functionality except for the basic API for further development. Our extensions to the ns2 code include:

- Modifications of the RTP and RTCP packet headers to include the fields for the feedback reports.
- Additional functionality for the generation of the RTCP sender and receiver reports.
- TCP-friendly bandwidth share estimations.
- Packet loss rate estimations.
- RTT estimations.
- Inter-arrival jitter estimations.

The interested reader can find more details on the ns2 extensions in our related work that is cited in [27].

4.1. Simulation environment and network topology setup

The topology that is used for the evaluation of the proposed protocol is a Local Area Network, which consists of one multimedia sender and six heterogeneous wired receivers. The heterogeneity of the receivers lays in the variation of the link capacity, which connects the receivers with the sender. We have intentionally created a ‘bottleneck’ between routers 1 and 2, in order to create two different sets of wired receivers. The first set of receivers (Nodes 1, 2, 3 ‘fast receivers’) is able to receive at higher bit rates than the second set (Nodes 4, 5, 6 ‘slow receivers’). By having the two sets of receivers with different receiving capabilities, we can investigate how the ASMP server would adjust the transmission rate based on the feedback reports from the slower receivers. The server transmits a single multicast stream to the set of all the wired receivers (fast and slow receivers) that join the session. The initial transmission rate was set to 150 kb/s. All receivers join the multicast stream at the same time. Figure 3 depicts the network topology for the simulated scenario.

We test ASMP under various simulation scenarios to investigate:

- The TCP-friendly behavior, when ASMP receivers share the same bottleneck link with multiple TCP connections.
- The smooth behavior.
- The reaction to a late-join event.
- The responsiveness to network changes due to congestion that is caused by other competing traffics.

As for the evaluation criteria, we considered the metrics described in [28].

4.2. *TCP-fairness*

In this simulation, we analyze the fairness toward competing TCP traffic without using the smooth function that is expressed in Equation (14). Node 7 (TCP Agent) starts transmitting TCP traffic to Node 8 (TCPSink), through the congested path from router 2 to router 3.

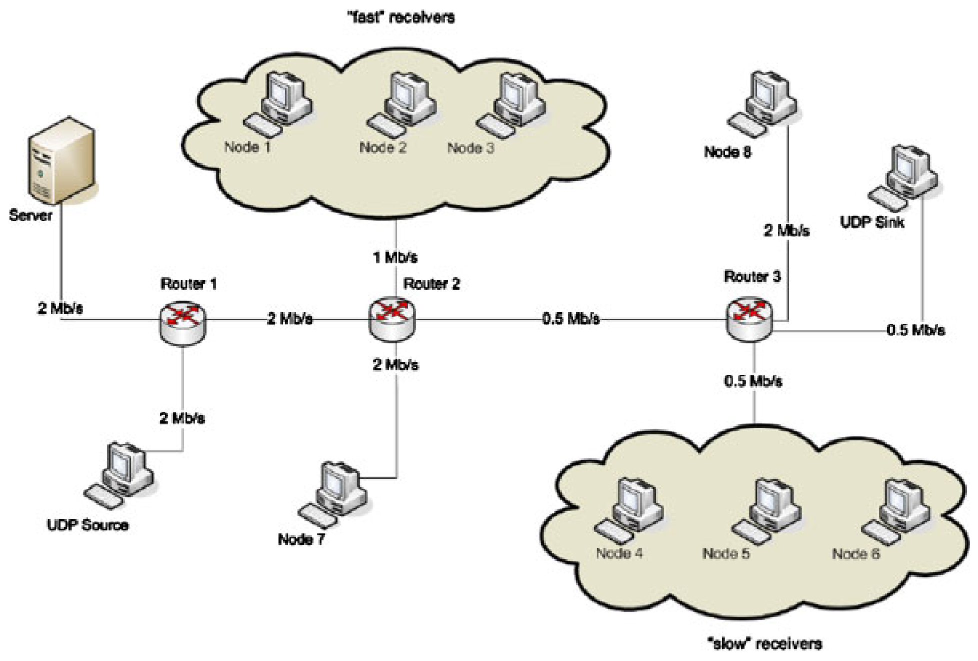

Figure 3. Simulated network topology.

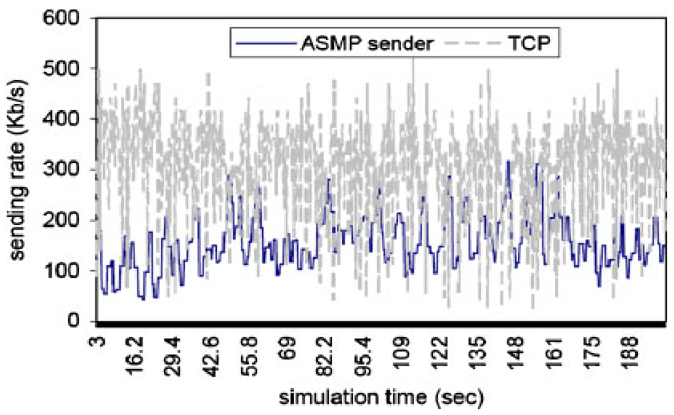

Figure 4. TCP-fairness—Sending rates.

We use in our evaluation Random Early Drop (RED) queue in the routers in order to avoid synchronization of the various data streams in the routers' queues. With this approach we ensure that all the transmitted streams will receive similar packet losses. A RED gateway detects upcoming 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.

As we observe in the simulation results (Figure 4), in the beginning of the simulation, the server backs off and reduces its transmission rate, as it encounters the first packet losses. In the remaining of the simulation time we observe that the TCP traffic is transmitted with even higher rates than the initial rate, confirming the TCP-friendly behavior of our congestion control protocol.

Fast and slow receivers enjoy the same receiving rates. However, our proposed solution is more than TCP-friendly that it ought to be. As we have previously explained, we try to integrate a TCP-friendly behavior based on RTCP sender and receiver feedback reports. The RTCP feedback reports are transmitted at higher intervals than TCP ACKs, making the ASMP sender react slowly to rapid networks changes.‡ TCP responses are faster than our protocol and as a result TCP occupies a larger portion of the available bandwidth.

However, we confirm that in the event of competing traffic the server transmits the multimedia data to the wired receivers at rates that provide acceptable receiving rates for multimedia data in terms of QoS. Figure 5 depicts the observed packet losses in the two representative nodes from the 'fast' and 'low' receiver's sets. Fast receivers present zero packet losses, whereas slow receivers present very low packet losses. Delay jitter in fast receivers is close to zero (it cannot even been projected in the simulation result chart), while in slow receivers the jitter values are between 1 to 23 ms (Figure 6).

‡According to [2] RTCP feedback reports interval is based on the number of receivers and as the number of receivers increases also the RTCP feedback reports interval increases. More information can be found in [2].

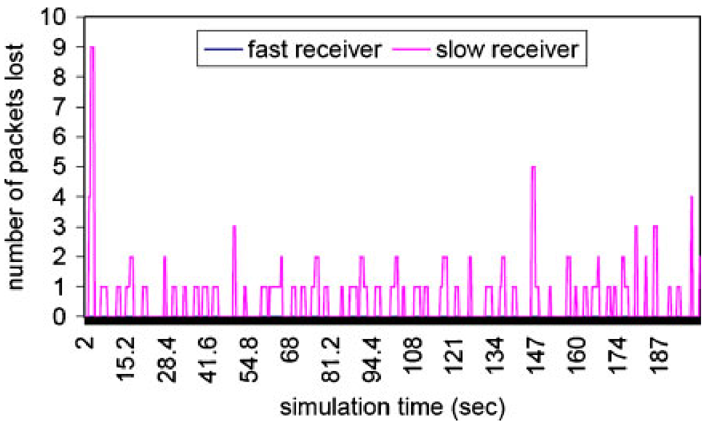

Figure 5. TCP-fairness—Packet losses.

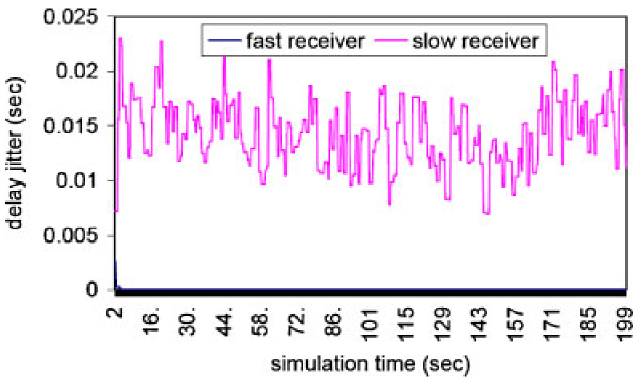

Figure 6. TCP-fairness—Delay jitter.

4.3. Smooth behavior

In this simulation we investigate whether or not our proposed solution for smooth transmission rates can indeed meet our design objectives, without challenging the transmission of TCP traffic. This simulation scenario has exactly the same network attributes as our previous simulation in order to achieve a fair comparison.

We observe from the simulation results (Figure 7) that TCP traffic enjoys high receiving rates. This is a first encouraged indication that a smooth adaptive transmission rate is a desired attribute not only for the multimedia server and the serving receivers but also for TCP traffic. We observe also in the same figure that the transmission rate of ASMP sender is smooth without seriously increasing packet losses (Figure 8). Fast receivers present zero packet losses. Therefore, we regard the amount of packets lost in slow receivers low for a multimedia application. Forward Error Correction (FEC) can address to this amount of packet losses. Delay jitter values are similar to the previous simulation (Figure 9). Fast receivers enjoy almost zero jitter delays.

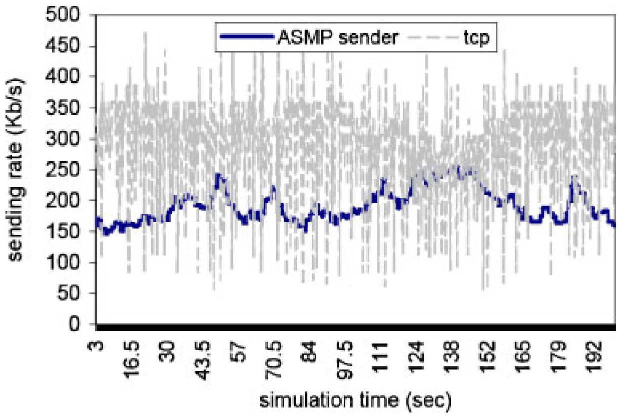

Figure 7. Smooth behavior—Sending rates.

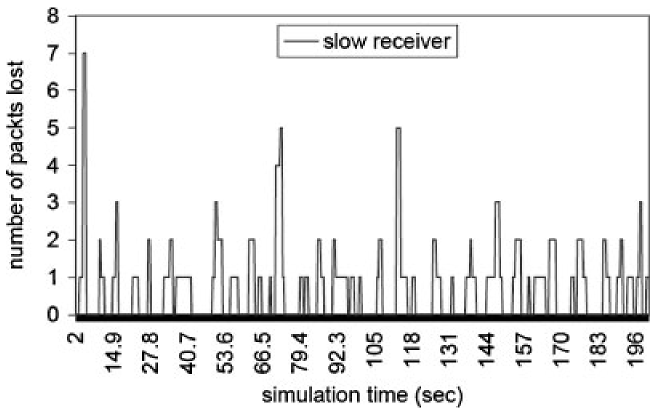

Figure 8. Smooth behavior—Packet losses.

Next in Figure 10 we present the comparison of the transmission rates with and without the smooth rate adaptation. It is clearly shown that smooth transmission rates prevent oscillations and as such they are more suitable for multimedia data transmission, due to the minimal distortion in AV encoding and decoding. The only drawback one can see is the slightest increase of packet losses due to the fact that the protocol cannot respond rapidly to the instantaneous network changes.

4.4. *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. 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 (Figure 11). In the beginning of the simulation, the ASMP sender transmits at a rate of

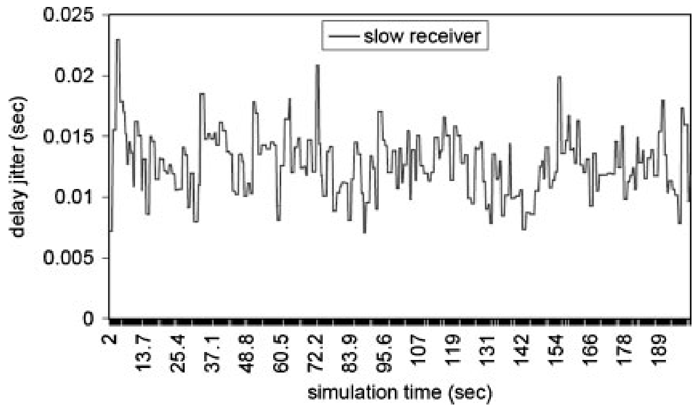

Figure 9. Smooth behavior—Delay jitter.

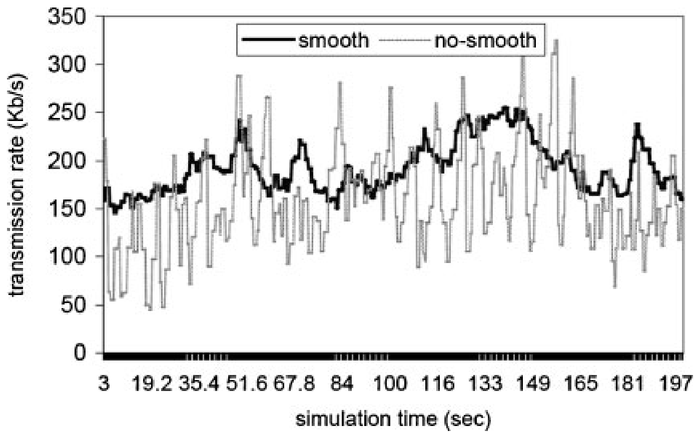

Figure 10. Smooth rate vs non-smooth.

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 300 kb/s link, joins the session. We observe that ASMP can handle this situation by adjusting the transmission rate with respect to the low receiver's bandwidth capacity (Figure 12). 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.

4.5. *Responsiveness to dynamic of competing traffic*

We test ASMP responsiveness to changes of competing traffic. We run a simple bottleneck simulation scenario in which the bottleneck link (1 Mb/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

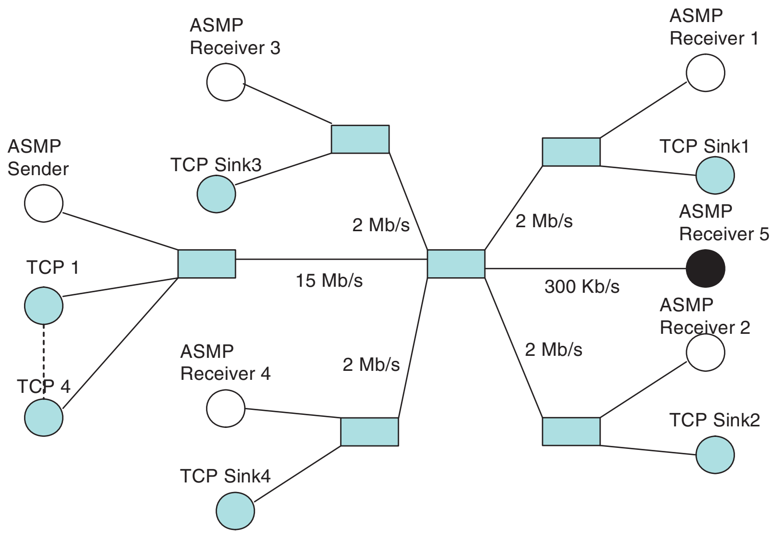

Figure 11. Late-join scenario topology.

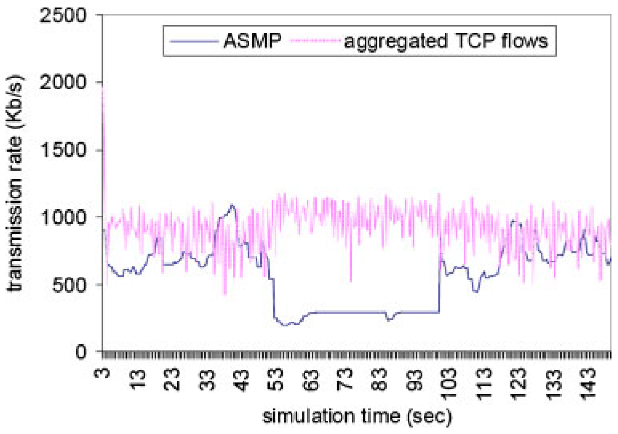

Figure 12. ASMP transmission rates with respect to late join of low-rate receiver.

with the ASMP stream. The second TCP flow (TCP2) starts its transmission at the 30th simulation time. Up to this point we have three flows in the bottleneck link. We observe in the simulation results (Figure 13) 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

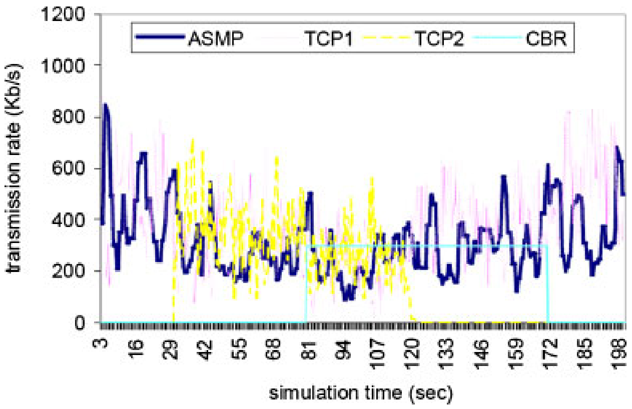

Figure 13. ASMP responsiveness to dynamics of competing traffic.

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.

5. COMPARISON WITH OTHER MULTICAST CONGESTION CONTROL MECHANISMS

In this section we present a comparison of ASMP and the most representative single multicast congestion available in the bibliography, TFMCC and PGMCC. Before the comparison we give a brief description of TFMCC and PGMCC.

5.1. TFMCC

TFMCC is a single-rate multicast congestion control protocol that extends TFRC from the unicast to multicast domain. The design goals of TFMCC are to provide a multicast congestion control that is TCP-friendly, high responsive to network changes and also suitable for multimedia data transmission. One important attribute of TFMCC is the feedback suppression algorithm in which only the receiver with the lowest receiving capacity, which is mentioned as the Current Limiting Receiver (CLR), sends frequent feedback reports. The sender adjusts its transmission rate based on CLR feedback reports. The rest of receivers, in the multicast group, send their feedbacks at higher time intervals in order to prevent feedback implosion phenomena. TFMCC's TCP-friendly bandwidth share is measured by receivers with the use of the following TCP equation [5]:

$$r_{\text{tcp}}^{i} = \frac{8s}{t_{\text{RTT}}\left(\sqrt{\frac{2p}{3}} + \left(12\sqrt{\frac{3p}{8}}\right) l(1+32p^2)\right)} \tag{21}$$

where r_{tcp}^{i} is the receiver's i estimation (in bytes/s), s is the packet size in bytes, p is the packet loss ratio, and t_{RTT} is the RTT of the link between the sender and the receiver. The long-term

TCP-friendliness is defined in TFMCC's specification to be no more than twice the sending rate of a TCP flow, which is traversing the same link with this TFMCC flow. A high-level overview of TFMCC functions is as follows:

- Each receiver measures the packet loss ratio based on the timestamps of packet headers.
- The receiver estimates the RTT in the path between itself and the sender.
- The receiver calculates a TCP-friendly bandwidth share with the use of Equation (21).
- A decision-making algorithm, which suppresses the feedback reports, defines the receivers that are eligible to transmit its feedback reports back to sender. Under this algorithm, only receivers with low transmission rates are allowed to send their feedback reports.
- The sender selects the receiver with the lowest receiving capacity as the CLR. The sending rate is adjusted to match the CLR's reported rate.

For the time being, TFMCC is under the review of the Internet community as an experimental RFC [5].

5.2. *PGMCC*

PGMCC is a single-rate multicast congestion control that is implemented as an extension of the Pragmatic Multicast Protocol (PGM) [29]. PGMCC uses a window-based TCP-like controller that is based on positive ACKs between the sender and the group representative. This representative (acker) is the receiver with the lowest receiving capacity in the multicast group. The selection of acker is based on NACKs that contain the loss rate as measured by receivers. NACKs are sent by all receivers in the multicast group, and the sender selects the acker with the following procedure:

Given operating parameter X_i (e.g. loss rate, RTT) for receiver i the sender computes the throughput $T(X_i)$ in order to determine the receiver with the lowest receiving capacity; the *acker*.

$$i : T(X_i) \leqslant T(X_j) \quad \forall j \in \{R\} \tag{22}$$

where R is the set of receivers in the multicast group.

PGMCC uses the following simplified form for the TCP-friendly bandwidth share:

$$r^i_{\text{tcp}} \propto \frac{1}{t^i_{\text{RTT}} \sqrt{p^i}} \tag{23}$$

where r^i_{tcp} is the sender's estimation for receiver i (in bytes/s), t^i_{RTT} is the RTT of the link between the sender and receiver i, and p^i is the packet loss rate of receiver i.

A high-level overview of PGMCC functionality is as follows:

- The sender computes the RTT time between itself and all receivers based on packet sequence numbers.
- The receivers send NACKs with the measured loss rate.
- The sender selects the group representative (acker) based on NACKs.
- The sender mimics a typical TCP connection between itself and the acker and adjusts the sending rate accordingly.

PGMCC measures the RTT between itself and all receivers in terms of packet numbers and not on timestamps and seems to be independent from synchronized clocks between sender and receivers. It is, however, depended on feedback reports from receivers because the sender needs

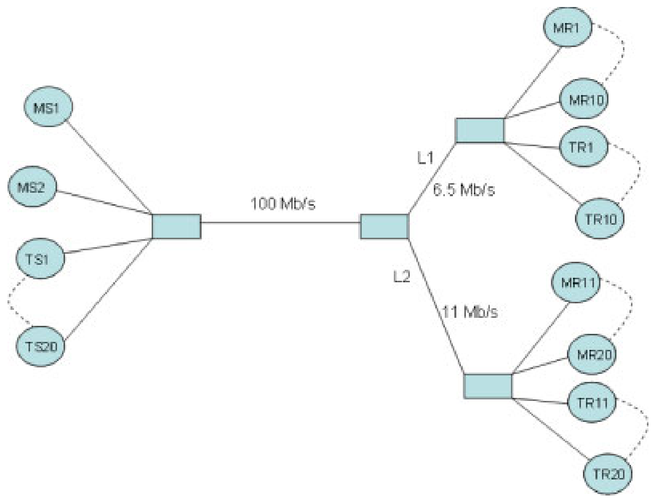

Figure 14. TCP-fairness topology.

to measure the RTT for all receivers to perform the selection of acker. Therefore, any feedback suppression algorithm would have serious effects on PGMCC's performance. In fact, PGMCC does not employ any feedback suppression mechanism.

5.3. Comparison with TFMCC

We implement TFMCC and ASMP in ns2 (source codes can be found in [30, 31], respectively) to evaluate its performance under a controlled environment. We run several simulations to investigate:

- The TCP-friendly behavior, when multicast receivers share the same bottleneck link with multiple TCP connections.
- The behavior of each congestion control when a 'slow' receiver late joins the multicast session.
- The responsiveness to rapid changes of the network conditions due to packet losses in the link.
- The responsiveness to dynamics of other competing data.

5.3.1. TCP fairness. In this simulation we evaluate the fairness of TFMCC and ASMP toward competing TCP traffic. We use a bottleneck scenario in which two multicast senders share multiple bottleneck links with 20 TCP Agents (Figure 14). MS and MR represent the multicast sender and receiver, whereas TS and TR stand for the TCP sender and receiver, respectively. The bottleneck links should be equally shared by multicast flow and TCP traffic, which means that the multicast receiver must not consume more than 9.09% of the bottleneck bandwidth (each bottleneck link, L1 and L2, is shared by 10 TCP connections and 1 multicast flow). As a result we expect each flow to receive $100\%/11 = 9.09\%$ of the bottleneck bandwidth. Therefore, multicast receivers in the low capacity link ($L1 = 6.5\,\text{Mb/s}$) must not consume more than 590.85 kb/s, whereas in the higher link ($L2 = 11\,\text{Mb/s}$) must not consume more than 1 Mb/s. The rest of the available bandwidth should be consumed by TCP connections and it is expected that each TCP connection will receive at least the same bandwidth share with the multicast flow.

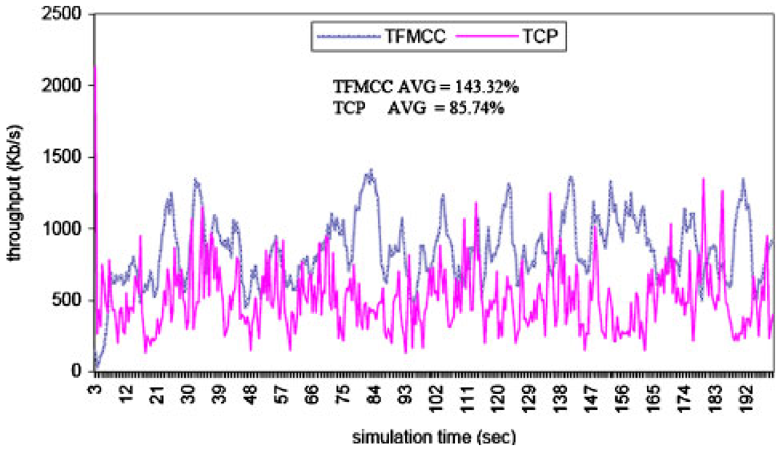

Figure 15. TFMCC vs TCP traffic in L1.

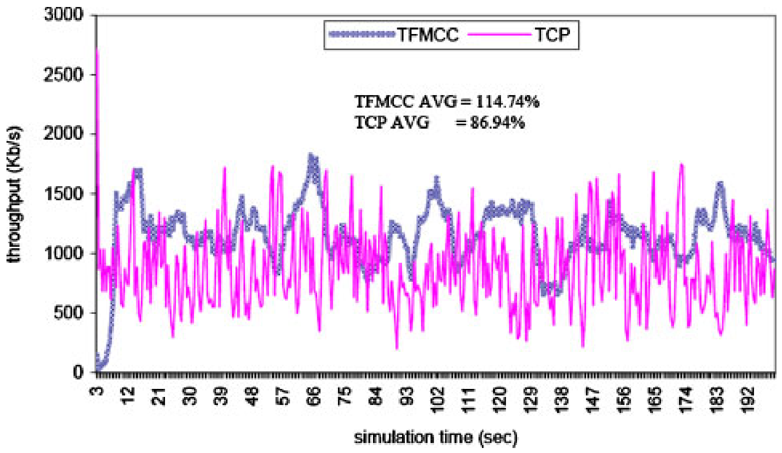

Figure 16. TFMCC vs TCP traffic in L2.

Figures 15 and 16 present the achieved throughput of TFMCC receivers versus TCP receivers in links L1 and L2, respectively. We observe that TFMCC in the low capacity bottleneck link (L1) consumes on average 143% of the expected bandwidth share while TCP is close to 85% of its share. In the higher capacity link (L2), TFMCC enjoys again higher bandwidth share than TCP connections. It is our assessment that TCP will suffer from starvation in the light of multiple TFMCC flows in Internet as TFMCC consumes much more bandwidth than a TCP flow when sharing the same bottleneck links.

On the other hand, ASMP is more TCP-friendly than TFMCC. We observe in Figures 17 and 18 that TCP enjoys a bandwidth share of 93% in both cases in L1 and L2, which is very close to absolute value of a TCP-friendly bandwidth share (100%). ASMP consumes less than its share but still high enough and close to 83% in the low capacity link.

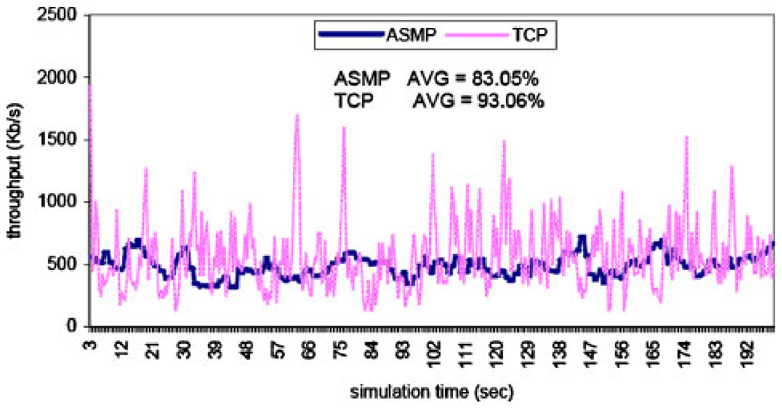

Figure 17. ASMP vs TCP traffic in L1.

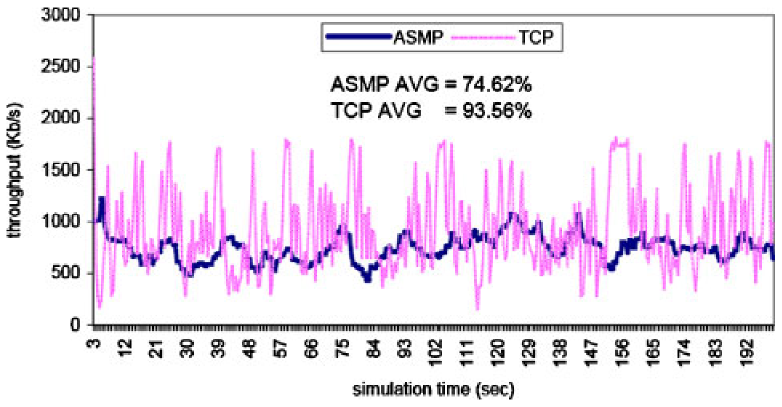

Figure 18. ASMP vs TCP traffic in L2.

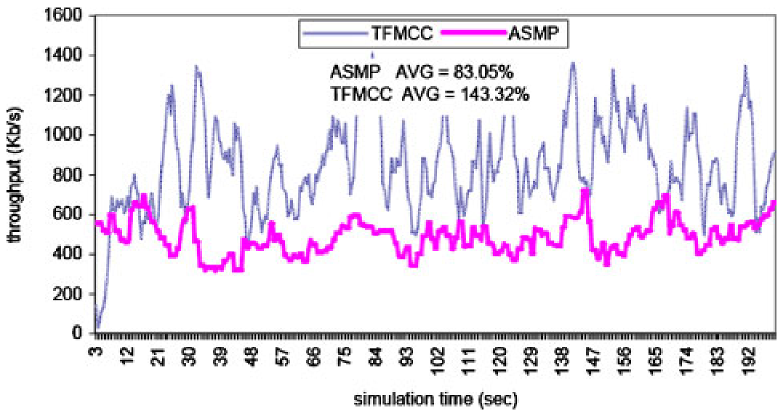

Figure 19. TFMCC vs ASMP bandwidth share in L1.

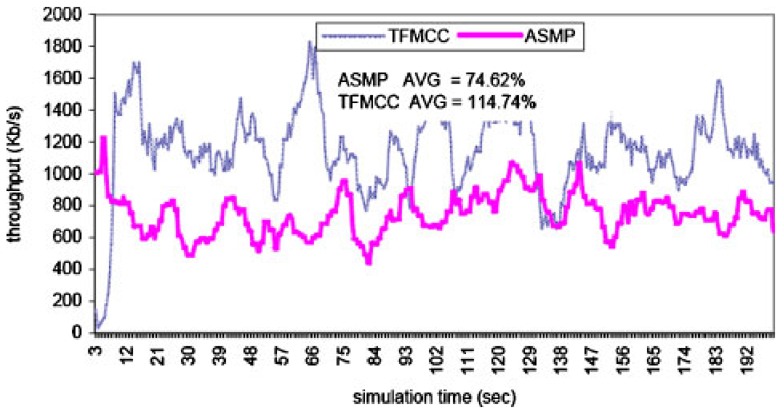

Figure 20. TFMCC vs ASMP bandwidth share in L2.

Table I. TFMCC vs TCP achieved throughput.

	Link utilization (%)	Average transmission rates
TFMCC	L1: 143.32	846 800.6 kb/s
	L2: 114.74	1.147 Mb/s
TCP	L1: 85.74	506.59 kb/s
	L2: 86.94	869.40 kb/s

Table II. ASMP vs TCP achieved throughput.

	Link utilization (%)	Average transmission rates (kb/s)
ASMP	L1: 83.05	490
	L2: 74.62	746.20
TCP	L1: 95.06	561.66
	L2: 93.56	935.60

However, except for TCP-friendliness we observe that ASMP is smoother than TFMCC, when we directly compare the transmission rates of each congestion control scheme in L1 and L2 (Figures 19 and 20). It is fair to say that ASMP presents much smoother transmission rates than TFMCC, which seems to suffer sometimes from high oscillations. A summary of the achieved throughputs is also presented in Tables I and II.

With this experiment we verify that the ASMP is not only more TCP-friendly than TFMCC but also smoother, which is a desired attribute for multimedia applications.

5.3.2. Late join of low-rate receiver. In this simulation scenario a bottleneck link (L1) with capacity of 1.5 Mb/s is shared by one multicast sender and four TCP connections (Figure 21). In the beginning of the simulation one multicast sender transmits at a rate of 500 kb/s and four receivers

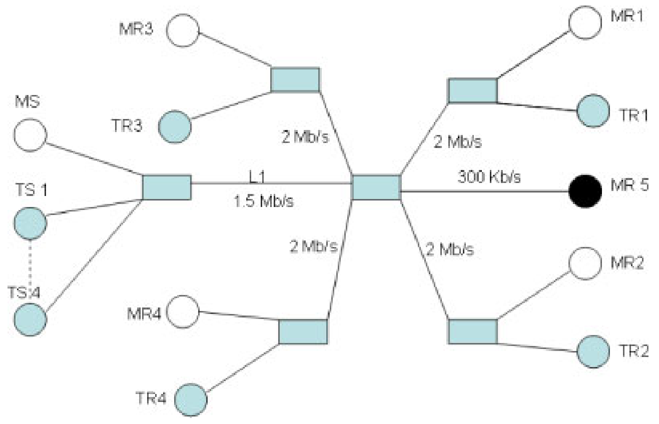

Figure 21. Late-join scenario topology.

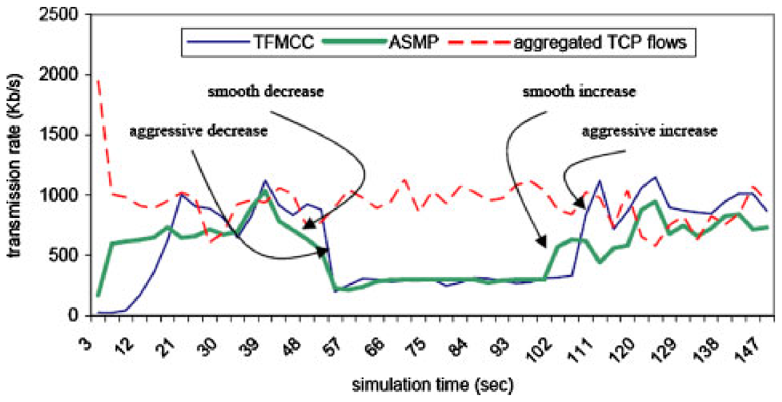

Figure 22. TFMCC vs ASMP transmission rates.

join the session. At the 50th simulation second a late receiver (MR5) with lower receiving capacity (300 kb/s) joins the multicast group. We observe that TFMCC needs few seconds to realize the presence of a low-capacity receiver before adjusting its transmission rate. In addition, TFMCC reacts aggressively by dropping immediately the transmission rate in the light of the first packet losses (Figure 22). In contrast, ASMP does not make abrupt changes in the transmission rate and adjusts its transmission rate as it is expected. At the 100th simulation second the low capacity receiver leaves the session and we observe that TFMCC increases again its transmission rate in such a way that creates high oscillations. ASMP starts gradually regaining its previous high transmission rates (Figure 22). Our conclusion from this experiment is that ASMP not only keeps a smooth transmission rate but also reacts rapidly to network topology changes.

5.3.3. Responsiveness to changes in the loss rate. In this simulation scenario we investigate how TFMCC and ASMP respond to changes in the loss rate and evaluate its performance. Loss rate

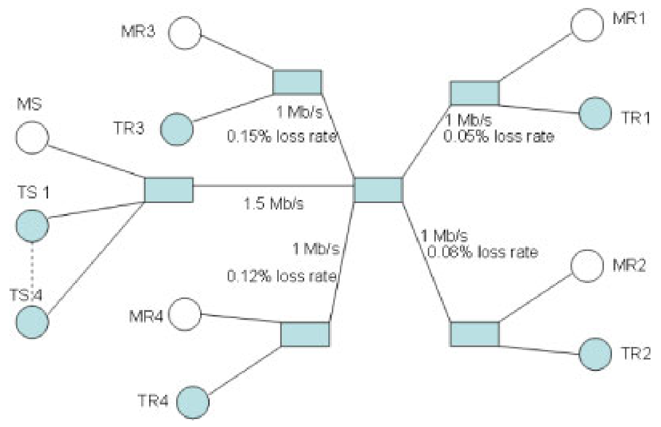

Figure 23. Varying loss rates topology.

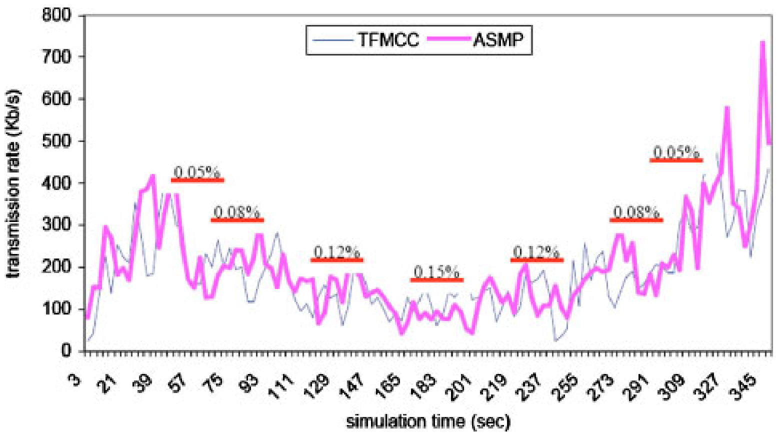

Figure 24. Responsiveness to varying loss rates.

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 (Figure 23). At the beginning of the simulation, we only have one receiver that joins the session while the rest of 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 s, 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 multicast traffic. Figure 24 depicts the simulation results. We observe that the smooth attribute does not seriously affect the responsiveness of ASMP. The reaction of TFMCC is very close to that of ASMP. We expected this behavior, as one important design goal of TFMCC is its smooth reaction to packet losses, so that it can be used for multimedia data transmission. Our implementation offers even smoother reactions in the light of packet losses than TFMCC.

5.3.4. Responsiveness to dynamics of competing traffic. In this simulation we compare TFMCC and ASMP in a heterogeneous dynamic network that was used for similar experimentations in [12]. In Figure 25 receivers R1–R8 are connected with 2 Mb/s links and there is at most one link with 100 ms delay between senders and receivers. On each link two TCP flows are randomly turned on and off according to Pareto distribution with the average value of 5 s. Two additional UDP flows of 400 kb/s on each link are also randomly turned on and off. These flows dynamically generate bottlenecks and make the network heterogeneous. At least there is a multicast flow from multicast sender (MS1) to receivers. Therefore, at every time instance there are at most five flows on any link: two TCP, two UDP and one multicast. The multicast flow is expected to receive approximately 400 kb/s bandwidth share. We observe in the simulation results that TFMCC (Figure 26) presents higher throughput than ASMP (Figure 27), while TCP throughput is almost the same (better when competing with ASMP) in both cases. Another interesting observation is that ASMP has high stability and adaptability in this heterogeneous environment, although we expected the smooth functions and high time intervals of RTCP reports would degrade its performance. However, it becomes clear that we need to increase the achievable throughput of ASMP in order to reach at least TCP throughput when both flows share the same bottleneck link (Figure 28).

5.4. Comparison with PGMCC

For the comparison between ASMP and PGMCC, we use simulation results from [8] and compare them with traces from identical simulations that we conducted with ns2.

5.4.1. TCP fairness. Figure 29 shows a simulation scenario involving two PGMCC receivers (MR1, MR2), which belong to the same flow, and one TCP receiver (TR). MS and TS stand for the multicast and TCP senders, respectively. Links L1 and L2 have capacity of 400 and 500 kb/s and

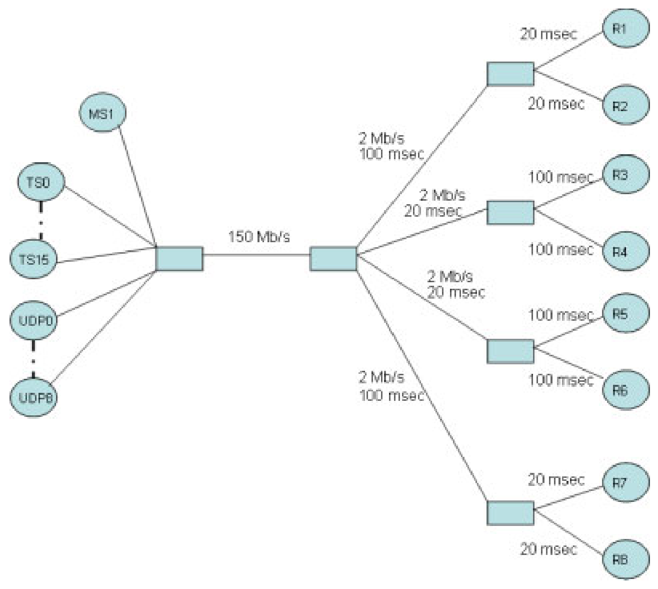

Figure 25. Heterogeneous dynamic network.

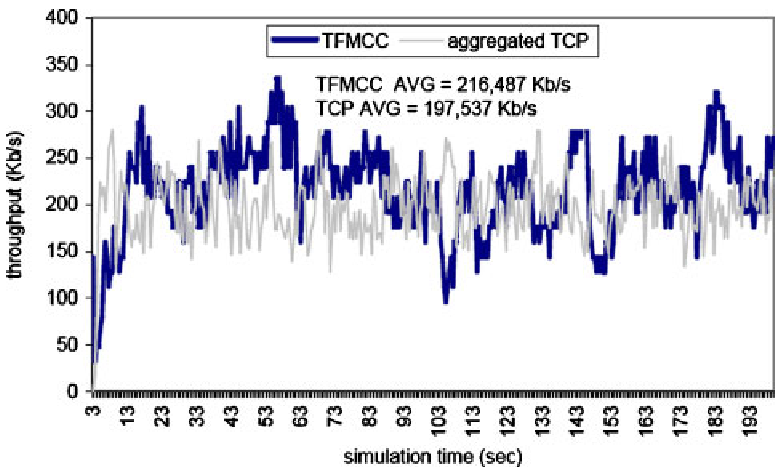

Figure 26. Responsiveness to competing traffic: TCP achieved throughput with competing UDP and TFMCC multicast flows.

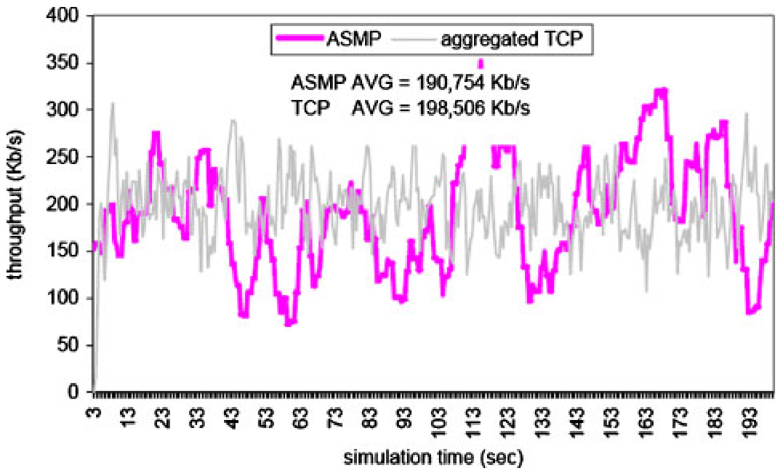

Figure 27. Responsiveness to competing traffic: TCP achieved throughput with competing UDP and ASMP flows.

the propagation delay for both links was set to 50 ms. In the beginning of the simulation receiver MR2 is started first, followed by MR1 and by TR.

We observe in Figure 30 that in the beginning of the simulation, PGMCC sender (MS) occupies the available bandwidth and when receiver MR2 joins the session (simulation time 17:04) it reduces its transmission rate to 400 kb/s. PGMCC sender further reduces its transmission rate down to approximately 220 kb/s when the TCP sender (TS) starts its transmission (simulation time 17:05). When the TCP sender terminates its transmission (simulation time 17:07) the PGMCC sender increases the session rate to 500 kb/s causing congestion to MR1. The congestion and packet

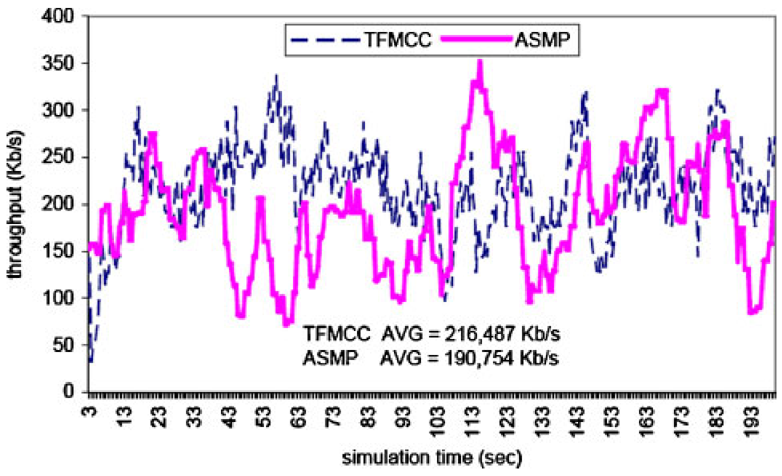

Figure 28. Responsiveness to competing traffic: TFMCC vs ASMP.

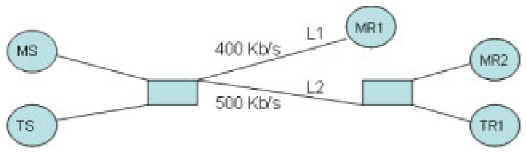

Figure 29. Bottleneck network.

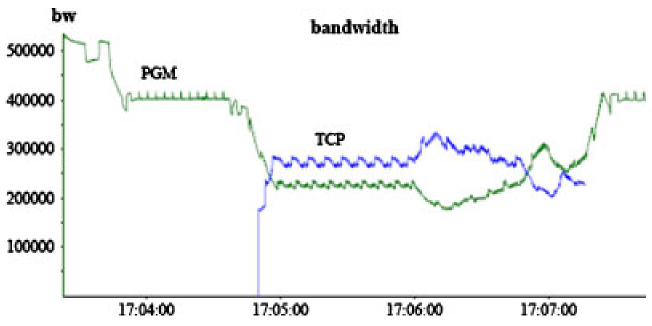

Figure 30. PGMCC vs TCP traffic.

losses result to an acker switch from MR2 to MR1 and the transmission rate is adjusted to a value around 400 kb/s. It is interesting that PGMCC presents stable behavior throughout the simulation. This mainly happens due to the fact that the acker switch occurs only between two PGMCC receivers. Our comment is that PGMCC transmission rate is indeed TCP-friendly, although in this mild-congested scenario PGMCC has lower throughput than TCP traffic.

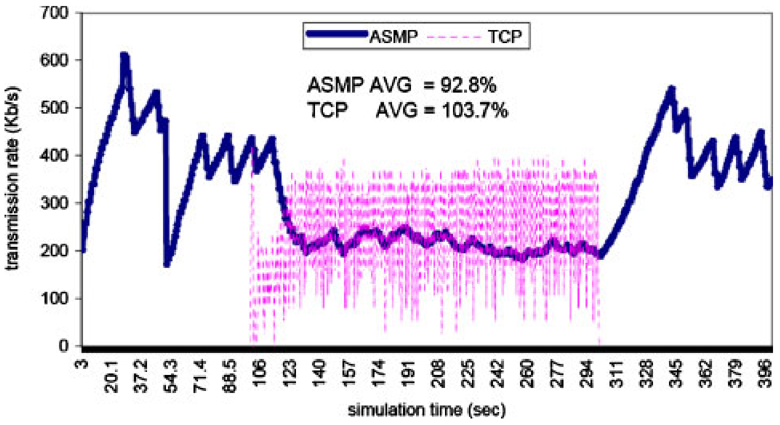

Figure 31. ASMP vs TCP traffic.

Figure 31 presents simulation results from the same topology with ASMP. We observe that in the beginning of the simulation, the ASMP sender tries to occupy all the available bandwidth. At the 50th simulation second, the ASMP receiver (MR1) joins the session and the sender reduces its transmission rate below 200 kb/s, based on TCP-friendly estimations of MR1. This happens because when a receiver first joins a session it has a low estimation of the available bandwidth. It takes the algorithm few seconds to converge and indeed at the 57th simulation second receiver MR1 calculates the available bandwidth to be around 400 kb/s. At the 100th simulation second MS further reduces its transmission rate as a result of congestion that is caused by TCP traffic. We observe that MS transmission rate is stable and smooth when link L2 is shared by multicast and TCP traffic. L2 is almost equally shared by the two sources. The MS sender has an average transmission rate of 232 068 kb/s and TCP 259 401 kb/s. When the TCP sender stops its transmission the multicast sender increases gradually its transmission rate in a smooth fashion, as it was expected.

We verify with this simulation scenario that ASMP presents almost the same results with PGMCC, although this scenario favors PGMCC in terms of the number of receivers in the group and the lack of dynamic competing traffic. It is our assessment that the frequent acker selection, as a result of a dynamically changed network, would reduce PGMCC's performance.

5.4.2. Responsiveness to losses. In this simulation we investigate PGMCC behavior and compare it with ASMP in a network topology with uncorrelated losses. We use a simulation scenario with 100 multicast receivers behind independent links with random loss ratio of 1% (Figure 32).

An additional link with same features is used for TCP traffic. In the beginning of the simulation 10 multicast receivers join the session followed by additional 90 receivers at the 300th simulation second. Figure 33 shows the simulation results of PGMCC. We observe that PGMCC achieves lower throughput than TCP, as PGMCC rate control is based on acker selection (one of the 100 PGMCC receivers). The presence of the additional 90 receivers does not seem, however, to affect PGMCC's performance.

Figure 34 shows simulation results from ASMP. TCP throughput is again higher than ASMP throughput, as TCP has higher responsiveness than the smooth congestion control in ASMP. The

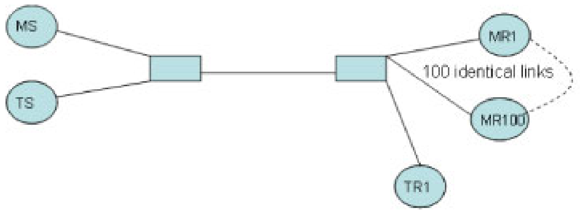

Figure 32. 100 Multicast Receivers, one TCP.

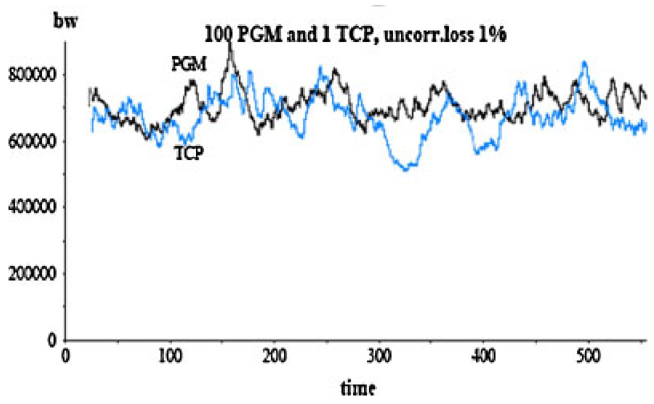

Figure 33. PGMCC responsiveness to mild losses.

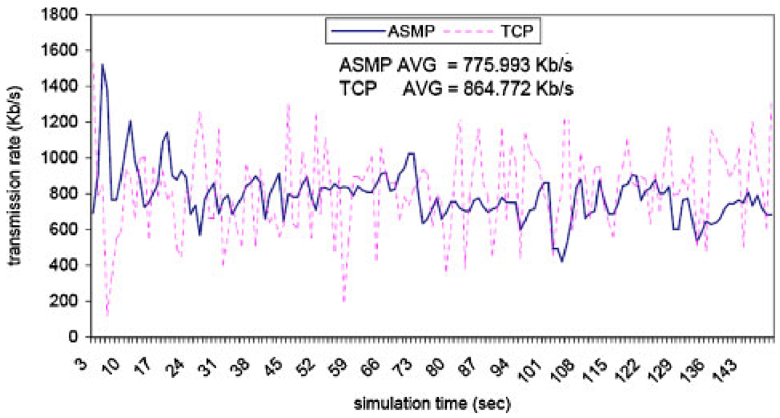

Figure 34. ASMP responsiveness to mild losses.

multicast sender's transmission rate drops at time 300 when the additional 90 receivers join the session. However, the transmission rate is adjusted again in few seconds to previous rates. The additional 90 receivers do not affect ASMP's performance.

We conclude from this medium-scale simulation that PGMCC and ASMP have similar behavior in terms of responsiveness to uncorrelated losses. Much larger-scale simulations/experiments are needed to further investigate the impact of losses and the scalability of ASMP.

6. CONCLUSIONS

We presented in this work ASMP, which is a new approach for the 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 measurements. 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.

We implemented ASMP in ns2 simulation software by including all the RTP/RTCP protocol's specification in RFC 3550, which are related to QoS metrics. Source code, simulation scripts, and results are available in [31].

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.

To compare the performance of ASMP against known related work we conducted simulations with TFMCC [5] and PGMCC [8]. Table III summarizes the comparison of ASMP with TFMCC and PGMCC. As the table shows, ASMP has a very good performance toward competing TCP traffic, while TFMCC seems to starve the TCP traffic. The TCP-friendliness of PGMCC is very close to that of ASMP.

TFMCC presents high fluctuations of the transmission rate, while ASMP has smooth and steady transmission rates in almost all cases. PGMCC's transmission rate is similar to ASMP in simulations involving mild losses and small number of receivers.

ASMP presents high responsiveness to packet losses and adapts very rapidly the changes in the network topology, although the convergence time is higher than that of TFMCC and PGMCC.

ASMP scalability is based on RTCP protocol's limitations in which only a small amount of the session bandwidth (5%) can be used for the dissemination of RTCP sender and receiver reports. In groups with large number of receivers the RTCP report intervals tend to increase so that ASMP cannot function properly. In such cases, additional feedback mechanisms have to be in place to ensure that the sender will receive feedback reports, especially from 'slow' receivers, in order to adjust its transmission rate. TFMCC presents high scalability with the use of suppression algorithms that select a group representative. PGMCC's scalability depends on network components. In any

Table III. Comparison of ASMP and other single-rate congestion control mechanisms.

	ASMP	TFMCC	PGMCC
TCP-friendliness	Very good	Poor	Good
Stable transmission rate	Very good	No	Good
Convergence time	Average	Good	Good
Scalability	Average (RTCP based)	Very good (group representative)	Average (requires NACs from all receivers)
Limitations	No	No	Requires support from network devices

case, PGMCC requires that all receivers in the group should send NACs so that the PGMCC sender can be able to select the group representative (acker).

7. FUTURE WORK

Apart from the observations made during the simulations with respect to the ASMP behavior, we believe that further studies of the 'smooth' transmission rate concept will benefit research in the area of multicast congestion control. We still need to investigate ASMP's scalability with a large number of receivers.

CI that are based on statistics of jitter delay measurements are very well fitted to wireless networks in which jitter delay measurements exploit better and more accurately the network congestion level than packet loss events.

Moreover, we will investigate deeper the effect of 'smoothens' on other competing traffic types and loss error schemes. It is also our intention to use our solution as part of a multi-rate transmission scheme. Finally, we will implement ASMP in real-world experiments and investigate its behavior with real multimedia traffic.

REFERENCES

1. Mankin A, Romanow A, Bradner S, Paxson V. IETF criteria for evaluating reliable multicast transport and application protocols. *RFC 2357*, June 1998.
2. Schulzrinne H, Casner S, Frederick R, Jacobson V. RTP: a transport protocol for real-time applications. *RFC 3550*, July 2003.
3. Pandhye J, Kurose J, Towsley D, Koodli R. A model based TCP-friendly rate control protocol. *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video* (*NOSSDAV*), Basking Ridge, NJ, June 1999.
4. Available from: http://www.isi.edu/nsnam/ns/.
5. Widmer J, Handley M. On TCP-friendly multicast congestion control (TFMCC). *RFC 4654*, 2006.
6. Widmer J, Handley M. Extending equation-based congestion control to multicast applications. *Proceedings of ACM SIGCOMM '01*, San Diego, U.S.A., 2001.
7. Handley M, Floyd S, Padhye J, Widmer J. TCP friendly rate control (TFRC). *RFC 3448*, Network Working Group, January 2003.
8. Rizzo L. pgmcc: a TCP-friendly single-rate multicast congestion control scheme. *Proceedings of the ACM SIGCOMM '00*, Stockholm, Sweden, 2000.
9. Smith H, Mutka M, Rover D. A feedback based rate control algorithm for multicast transmitted video conferencing. *Journal of High Speed Networks* 1998; **7**(3–4):259–279.
10. Sisalem D, Wolisz A. LDA+TCP-friendly adaptation: a measurement and comparison study. *Proceedings of the 10th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (*NOSSDAV'2000*), Chapel Hill, NC, U.S.A., 25–28 June 2000.
11. Macker J, Adamson R. A tcp friendly, rate-based mechanism for nack-oriented reliable multicast congestion control. *Proceedings of IEEE GLOBECOM*, San Antonio, TX, U.S.A., 2001.
12. Jiang L, Yuksel M, Kalyanaraman S. Explicit rate multicast congestion control. *Computer Networks*: *The International Journal of Computer and Telecommunications Networking* 2006; **50**(15):2614–2640.
13. McCanne S, Jacobson V, Vetterli M. Receiver-driven layered multicast. *Proceedings of the ACM SIGCOMM*, Stanford University, CA, U.S.A., 1996.
14. Fenner W. Internet Group management Protocol, version 2. *RFC 2236*, Network Working Group, November 1997.

15. Legout A, Biersack E. PLM: fast convergence for cumulative layered multicast transmission schems. *Proceedings of the ACM SIGMETRICS*, Santa Clara, CA, U.S.A., 2000.
16. Vicisiano L, Rizzo L, Crowcroft J. TCP-like congestion control for layered multicast data transfer. *IEEE INFOCOM*, San Francisco, U.S.A., March 1998; 996–1003.
17. Byers JW *et al.* FLID-DL congestion control for layered multicast. *Proceedings of the NGC*, Palo Alto, CA, U.S.A., 2000.
18. Byers J, Luby M, Mitzenmacher M. Fine-grained layered multicast. *Proceedings of the INFOCOM 2001*, Alaska, U.S.A., vol. 2, 2001; 1143–1151.
19. Byers J, Kwon G. STAIR: Practical aimd multirate multicast congestion control. *Proceedings of the NGC*, London, U.K., 2001.
20. Kwon G-I, Byers J. Smooth multirate multicast congestion control. *Proceedings of the IEEE INFOCOM*, San Francisco, U.S.A., 2003.
21. Li J, Kalyanaraman S. Generalized multicast congestion control. *Proceedings of the 5th COST 264 International Workshop on Network Group Communications* (*NGC 2003*) *and ICQT*, Munich, Germany, 2003.
22. Gharai L. RTP with TCP friendly rate control. draft-ietf-avt-tfrc-profile-07 (work in progress), March 2007.
23. Bouras C, Gkamas A, Kioumourtzis G. Smooth multicast congestion control for adaptive multimedia transmission. *NGI 2008*, Krakow, Poland, April 2008.
24. Bouras C, Gkamas A. Streaming multimedia data with adaptive QoS characteristics. *Fifth International Conference on Protocols for Multimedia Systems—PROMS 2000*, Cracow, Poland, 22–25 October 2000; 129–139.
25. Hamto F, Sirisena H. Cumulative inter-ADU jitter concept and its applications. *Proceedings of the IEEE ICCCN*, Scottdale, AZ, October 2001; 531–534.
26. Nonnenmacher J, Ernst Biersack W. Optimal multicast feedback. *Proceedings of the Conference on Computer Communications* (*IEEE Infocom*), San Francisco, U.S.A., March 1998.
27. Bouras C, Gkamas A, Kioumourtzis G. 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.
28. Floyd S. Metrics for the evaluation of congestion control mechanisms. *RFC 5166*, March 2008.
29. Speakman T *et al.* PGM reliable transport protocol specification. *RFC 3208*, December 2001.
30. Available from: http://www.informatik.uni-mannheim.de/pi4/projects/congctrl/tfmcc/installation.html (accessed on 5 January 2008).
31. Available from: http://ru6.cti.gr/ru6/ns_rtp_home.php (accessed on 29 February 2008).

AUTHORS' BIOGRAPHIES

Christos Bouras obtained his Diploma and PhD from the Department Of Computer Engineering and Informatics of Patras University (Greece). He is currently a Professor in the above department. In addition, he is a scientific advisor of Research Unit 6 in Research Academic Computer Technology Institute (CTI), Patras, Greece. His research interests include the Analysis of the Performance of Networking and Computer Systems, Computer Networks and Protocols, Telematics and New Services, QoS and Pricing for Networks and Services, e-Learning Networked Virtual Environments and WWW Issues. He has extended his professional experience in Design and Analysis of Networks, Protocols, Telematics and New Services. He has published 300 papers in various well-known refereed conferences and journals. He is a co-author of eight books in Greek. He has been a PC member and referee in various international journals and conferences. He has participated in R&D projects such as RACE, ESPRIT, TELEMATICS, EDUCATIONAL MULTIMEDIA, ISPO, EMPLOYMENT, ADAPT, STRIDE, EUROFORM, IST, GROWTH and others. In addition, he is a member of experts in the Greek Research and Technology Network (GRNET), Advisory Committee Member to the World Wide Web Consortium (W3C), IEEE-CS Technical Committee on Learning Technologies, IEEE ComSoc Radio Communications Committee, IASTED Technical Committee on Education WG6.4 Internet Applications Engineering of IFIP, ACM, IEEE, EDEN, AACE, New York Academy of Sciences and Technical Chamber of Greece.

Apostolos Gkamas obtained his Diploma, Master Degree and PhD from the Computer Engineering and Informatics Department of Patras University (Greece). He is currently an R&D Computer Engineer at the Research Unit 6 of the Research Academic Computer Technology Institute, Patras, Greece. He is also a visiting lecturer in the Hellenic Open University. His research interests include Computer Networks, Telematics, Distributed Systems, Multimedia and Hypermedia. More particularly, he is engaged in the transmission of multimedia data over networks and multicast congestion control. He has published more than 40 papers in international Journals and well-known refereed conferences. He is also co-author of three books (one with subject Multimedia and Computer Networks one with subject Special Network Issues and one with subject IPv6). He has participated in various R&D project (in both EU and national) such as IST, FP6, eLearning, PENED, EPEAEK, Information Society.

Georgios Kioumourtzis is a PhD candidate in the Computer Engineering and Informatics Department of Patras University (Greece). He is currently a Lieutenant Colonel in the Greek Army. He received his BS from the Hellenic Military Academy and graduated also from the School of Telecommunications for Signal Officers. In 2005, he received the Master of Science in System's Engineering and the Master of Science in Computer Science from the Naval Postgraduate School (NPS) California, U.S.A. His thesis work was related to Mobile *Ad Hoc* Wireless Networks (MANETs). His current research interests include Multimedia transmission over heterogeneous networks, transmission protocols, cross-layer optimization in wireless networks and IEEE 802.11x technology.