

<i>prepared by</i> C.Burmeister, T.Klinner	<i>phone</i> +49(0)6103 766-263	
<i>approved by</i> Rolf Hakenberg	<i>version</i> A	<i>date</i> 30.10.2001

to:

cc:

# Low Delay Feedback RTCP - Timing Rules Simulation Results -



## Table of Contents

1	Introduction	3
2	The New RTP Profile – AVPF	4
3	Simulation Environment	6
3.1	Network Simulator Version 2	6
3.2	RTP Agent	6
3.3	Scenarios	6
3.4	Topologies	6
3.4.1	Unicast Topology (T-2)	6
3.4.2	Small Multicast Topology (T-4)	7
3.4.3	Medium Size Multicast Topology (T-8)	7
3.4.4	Large Size Multicast Topology (T-16)	8
4	RTCP Bit Rate Measurements	9
4.1	Unicast Topology	9
4.1.1	No Packet Losses	9
4.1.2	Packet Losses with a Constant Loss Rate	10
4.2	Multicast Topologies	11
4.2.1	No Packet Losses	13
4.2.2	Packet Losses with a Constant Loss Rate	15
5	Feedback Measurements	16
5.1	Unicast Simulations	16
5.2	Multicast Simulations	17
5.2.1	Shared Losses vs Distributed Losses	18
5.2.2	Sender vs Receiver	19
6	Investigations on “k”	21
6.1	Probability of Feedback Suppression	21
6.2	Sample Scenarios	23
6.3	Feedback Suppression in the Sample Scenarios	23
6.4	Loss Report Delay	26
6.5	Summary of “k”-Investigations	26
7	Investigations on “l”	28
7.1	Probability of Feedback Suppression	28
7.2	Feedback Suppression in the Sample Scenarios	28
7.3	Loss Report Delay	30
7.4	Summary of “l”-Investigations	31
8	Summary	32
	Bibliography	33
	Annex A: Probability Functions	34

---

## 1 Introduction

The Real-time Transport Protocol (RTP) is widely used for the transmission of real-time or near real-time media data over the Internet. While it was originally designed to work well for multicast groups in very large scales, its scope is not limited to that. More and more applications use RTP for small multicast groups (e.g. video conferences) or even unicast (e.g. media streaming applications).

RTP comes together with its companion protocol Real-time Transport Control Protocol (RTCP), which is used to monitor the transmission of the media data and provide feedback of the reception quality. What's more it can be used for a loosely session control. Having the scope of large multicast groups in mind, the rules when to send feedback were much restricted to avoid feedback explosion or feedback related congestion in the network. RTP and RTCP have proven to work well in the Internet, especially in large multicast groups, which is shown by its tremendous usages today.

However the applications that transmit the media data only to small multicast groups or unicast, may benefit from more frequent feedback. The source of the packets might be able to react to changes in the reception quality, which might be due to congestion in the network or other sudden changes. Possible reactions include sending rate adaptation according to a congestion control algorithm or the invocation of error resilience features for the media stream (e.g. retransmissions, reference picture selection, NEWPRED, etc.).

As said before, more feedback would be needed to increase the reception quality, but RTP restricts the use of RTCP feedback very much. Hence it was decided to create a new extended RTP profile, which redefines some of the RTCP timing rules, but keeps most of the algorithms for RTP and RTCP, which have proven to work well. The new rules should scale from unicast to multicast, where unicast or small multicast applications have the most gain from it. A detailed description of the new profile and its timing rules can be found in [AVPF].

This document investigates the new algorithms by the means of simulations. We show that the new timing rules scale and behave network friendly. Therefore we first describe roughly the key features of the new RTP profile in Section 2. After that we describe the environment that is used to conduct the simulations in Section 3. Section 4 describes simulation results that show the backwards compatibility to RTP and that the new profile is network friendly. In Section 5 we show the benefit that applications could get from implementing the new profile and investigate some detailed questions.

## 2 The New RTP Profile – AVPF

As said above, RTP restricts the usage of RTCP feedback. The main rules that restrict the feedback are as follows:

1. RTCP messages are sent in compound packets, i.e. every RTCP packet contains at least one sender report (SR) or receiver report (RR) message and a source description (SDES) message.
2. The RTCP compound packets are sent in time intervals ( $T_{rr}$ ), which is computed as a function of the average packet size, the number of senders and receivers in the group and the session bandwidth. (-> 5% of the session bandwidth is used for RTCP messages; this bandwidth is shared between all session members, where the senders might get more than the receivers.)
3. The minimum interval between two RTCP packets from the same source is 5 seconds.

We see that these rules prevent feedback explosion and scale to very large multicast groups. However they do not allow timely feedback at all. While the second rule scales also to small groups or unicast (in this cases the interval might be as small as a few milliseconds), the third rule prevents the receivers from sending feedback in time.

The timing rules to send RTCP feedback from the new RTP profile (AVPF) consists of two key components. First the minimum interval of 5 seconds is abolished. Second, receivers get once during their (now quite small) RTCP interval the chance to send an RTCP packet “early”, i.e. not according to the calculated interval, but virtually immediately. It is important to note that the RTCP interval calculation is still inherited from the original RTP specification.

During normal operation, all group members calculate the RTCP interval ( $T_{rr}$ ) according to the rules, specified in RTP, besides that the 5 second minimum interval is not used. In these intervals, the members send normal RTCP compound packets, i.e. containing RR or SR, SDES and optional other messages. These packets give already congestion control algorithms the possibility to work properly, because the feedback is already quite frequent for small groups or unicast.

In case a receiver of a media stream feels the need to send feedback as soon as possible, e.g. a packet loss was detected, it may send an “early packet”. This packet might be sent before the time for which the next regularly scheduled packet should have been sent. The algorithm for scheduling the early packet is as follows:

The receiver feels the need to send feedback at time  $t_0$ . It checks if the last RTCP packet it has sent was already an “early packet” or if another “early packet” is already scheduled for transmission. If so it is not allowed to send another one, but may append the feedback it has to send to the next scheduled RTCP packet. If not it schedules an early packet. Therefore it calculates a maximum dithering interval,  $T_{dither\_max}$ . It is calculated as follows:

- i) If the session is a unicast session (group size = 2) then  
 $T_{dither\_max} := 0$ .
- ii) If the receiver has an RTT estimate to the originator of the media unit to provide feedback about, then  
 $T_{dither\_max} := k * T_{rtt} / 2 * members$ ,  
with  $k=1$ ,  $T_{rtt}$  indicating the round trip time.
- iii) If the receiver does not have an RTT estimate to the originator, then  
 $T_{dither\_max} := l * T_{rr}$ ,  
with  $l=0.5$ . and  $T_{rr}$  indicating the RTCP interval.

If the next regularly scheduled RTCP packet is within the interval from  $t_0$  to  $(t_0 + T_{dither\_max})$  no early packet is scheduled, but the feedback is appended to the next RTCP packet. Else a dithering interval  $T_{dither}$  is chosen uniformly distributed randomly from  $[0; T_{dither\_max}]$ . The early packet is scheduled for  $t_e = t_0 + T_{dither}$ .

While waiting for the early packet to be sent, the receiver may receive a RTCP feedback packet from another session member. This packet is checked for feedback messages. If it contains a feedback message,

---

which has the same content as some feedback that is to be sent, the feedback is stripped off the scheduled RTCP packet. If an early packet was sent, which has no feedback messages anymore, because the feedback content was already sent from a different group member, the early packet is cancelled and the original RTCP packet is scheduled again.

Early packets are also RTCP compound packets, but may not include a SDES message. Thus it may only consist of a SR or RR and the feedback message(s).

After sending the early packet, a flag `allow_early` is set to false, in order to prevent the same source from sending two consecutive early packets. After the next regularly scheduled RTCP packet was sent, the parameter would be set to true again. The next RTCP packet is scheduled for  $tn = tp + 2 * T_{rr}$ , where  $tn$  is the time of the next RTCP packet,  $tp$  the time of the last regularly scheduled one and  $T_{rr}$  the RTCP interval.

If another event should be reported before the next regularly scheduled RTCP packet was sent, the receiver is not allowed to send an early packet. However, if the time to the next regularly scheduled RTCP packet is less than the parameter `T_max_fb_delay`, it may append the feedback to the RTCP packet. Else it must discard the feedback.

A detailed description of the timing rules, the algorithms and feedback messages can be found in [AVPF].

## 3 Simulation Environment

This section describes the simulator that was used for the investigations and its key features. The extensions to the simulator, that were necessary are described roughly.

### 3.1 Network Simulator Version 2

The simulations were conducted using the network simulator version 2 (ns2). ns2 is an open source project, written in a combination of Tool Command Language (TCL) and C++. The scenarios are set-up using TCL. In the scripts it is possible to specify the topologies (nodes and links, bandwidths, queue sizes or error rates for links) and the parameters of the “agents”, i.e. protocol configurations. The protocols itself are implemented in C++ in the agents, which are connected to the nodes. A detailed description of ns2 and a downloadable newest version can be found at [ns2-homepage].

### 3.2 RTP Agent

We implemented a new agent, based on RTP/RTCP. RTP packets are sent at a constant packet rate with the correct header sizes. RTCP packets are sent according to the timing rules of [RTP] and also its algorithms for group membership maintenance are implemented. Sender and receiver reports are sent and the senders use these reports to maintain a RTT estimation to the other group members, as it is described in [RTP].

Further we extended the agent to support the new AVPF. The use of the new timing rules can be turned on and off via parameter settings in TCL.

### 3.3 Scenarios

The scenarios that are simulated are defined in TCL scripts. We set-up several different topologies, ranging from unicast with two session members to multicast with up to 25 session members. Depending on the used sending rates and the corresponding link bandwidths congestion losses may occur. In some scenarios, bit errors are inserted on certain links. We simulated groups with RTP/AVP agents, RTP/AVPF agents and mixed groups.

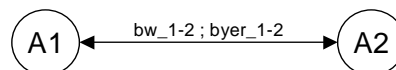
The feedback messages are generally NACK messages as defined in [AVPF] and are triggered by packet loss.

### 3.4 Topologies

Four different topologies are simulated, differing in the number of session members.

#### 3.4.1 Unicast Topology (T-2)

The topology is depicted below.



**Figure 1: Unicast Topology – T-2**

The agents A1 and A2 can have independent protocol configurations, .e.g. one or both of them could be sender or receiver. They are connected by a symmetric duplex link with a bandwidth of bw\_1-2 and a byte error rate of byer\_1-2.

### 3.4.2 Small Multicast Topology (T-4)

The small multicast topology consists of four session members. All connections are symmetric and duplex links. It is shown below:

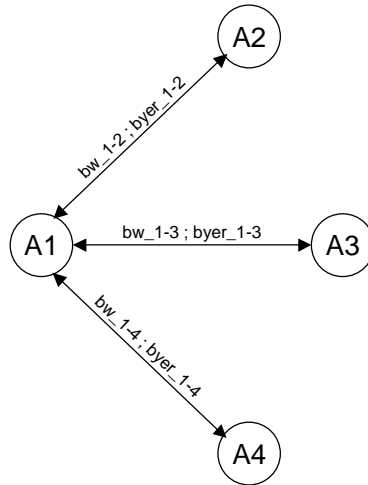


Figure 2: Small Multicast Topology – T-4

### 3.4.3 Medium Size Multicast Topology (T-8)

This topology consists of eight agents, which are connected again via symmetric duplex links as shown below:

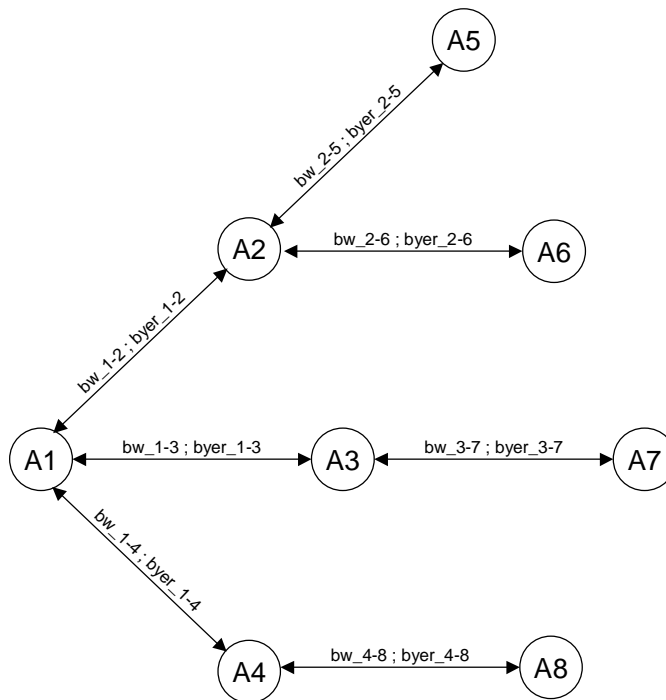


Figure 3: Medium Size Multicast Topology – T-8

### 3.4.4 Large Size Multicast Topology (T-16)

This is the largest topology we simulated for the normal investigations.

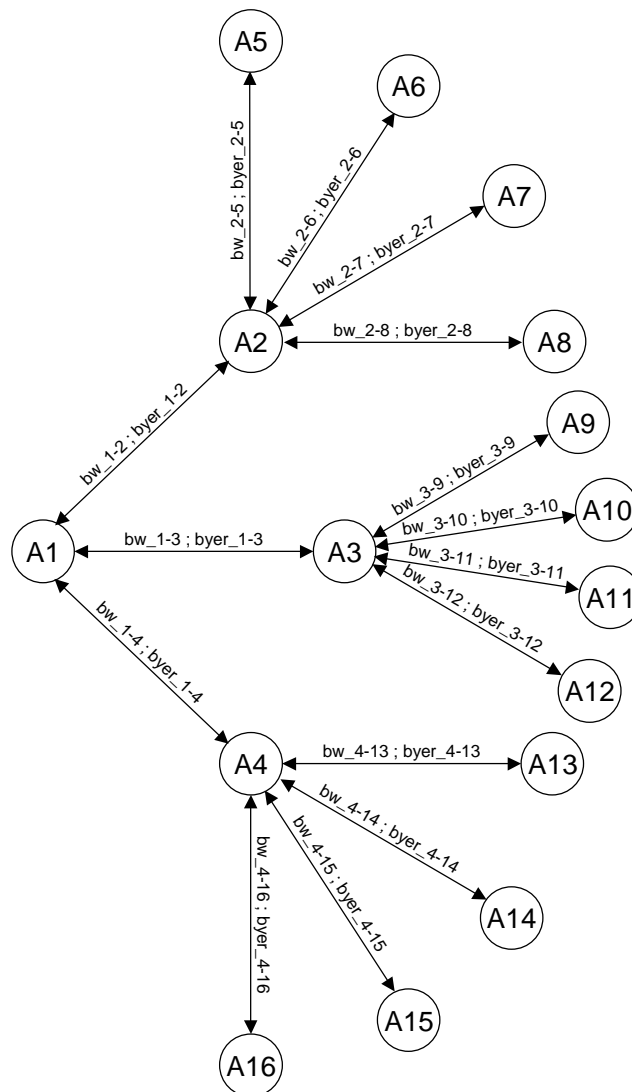


Figure 4: Large Size Multicast Topology - T-16

## 4 RTCP Bit Rate Measurements

The new timing rules allow more frequent RTCP feedback for small multicast groups. In large groups the algorithm behaves similar to usual RTP. While it is generally good to have more frequent feedback it cannot be allowed at all to increase the bit rate used for RTCP above a fixed limit, i.e. 5% of the total RTP bandwidth according to RTP. This section shows that with the new timing rules we keep the 5% limit for all investigated scenarios, topologies and group sizes. What is more, we show that mixed groups, i.e. some members use AVP some use AVPF, can be allowed and that each session member behaves fair according to its corresponding specification.

### 4.1 Unicast Topology

First we measured the RTCP bandwidth share in the unicast topology T-2. Even for a fixed topology and group size, there are several protocol parameters which are varied to simulate a large range of different scenarios. The different configurations are explained with the following table.

Configuration Number	Session Bandwidth	Agents that are RTP senders	Agents that are only receivers	Agents using AVP	Agents using AVPF
T2-1	2 Mbps	1	2	-	1,2
T2-2	2 Mbps	1,2	-	-	1,2
T2-3	2 Mbps	1	2	1	2
T2-4	2 Mbps	1,2	-	1	2
T2-5	2 Mbps	1	2	1,2	-
T2-6	2 Mbps	1,2	-	1,2	-
T2-7	200 kbps	1	2	-	1,2
T2-8	200 kbps	1,2	-	-	1,2
T2-9	200 kbps	1	2	1	2
T2-10	200 kbps	1,2	-	1	2
T2-11	200 kbps	1	2	1,2	-
T2-12	200 kbps	1,2	-	1,2	-
T2-13	20 kbps	1	2	-	1,2
T2-14	20 kbps	1,2	-	-	1,2
T2-15	20 kbps	1	2	1	2
T2-16	20 kbps	1,2	-	1	2
T2-17	20 kbps	1	2	1,2	-
T2-18	20 kbps	1,2	-	1,2	-

**Table 1: Configurations of the Unicast Simulations**

#### 4.1.1 No Packet Losses

First we consider scenarios where no losses occur. In this case both RTP session members transmit the RTCP compound packets at regular intervals, calculated as  $T_{rr}$ , if they use the AVPF, and use the minimum interval of 5s if they implement the AVP. No early packets are sent, because the need to send feedback is not given. Still it is important to see that not more than 5% of the session bandwidth is used for RTCP and AVP and AVPF members can co-exist.

Figure 5 shows the RTCP bandwidth share for the configurations of Table 1. We can see that in configurations, where both Agents use the new timing rules (T2-1, T2-2, T2-7, T2-8, T2-13 and T2-14) each of them uses about 2.5% of the session bandwidth for RTP, which sums up to 5% of the session bandwidth for both. This is achieved regardless of the agent being a sender or a receiver.

In the cases where Agent 1 uses AVP and Agent 2 AVPF, the total RTCP session bandwidth is decreased. This is due to the fact that Agent 1 can send RTCP packets only with a minimum interval of 5 seconds. Thus only a small fraction of the session bandwidth is used for its RTCP packets. For a high bit rate session (session bandwidth = 2 Mbps) the fraction of the RTCP packets from Agent one is as small as 0.01%. For smaller session bandwidths the fraction increases, because the same amount of RTCP data is sent. The bandwidth share that is used by RTCP packets from Agent 2 is not different from what was used, when both Agents implemented the AVPF. Thus the interaction of AVP and AVPF agents is not problematic in this scenarios at all.

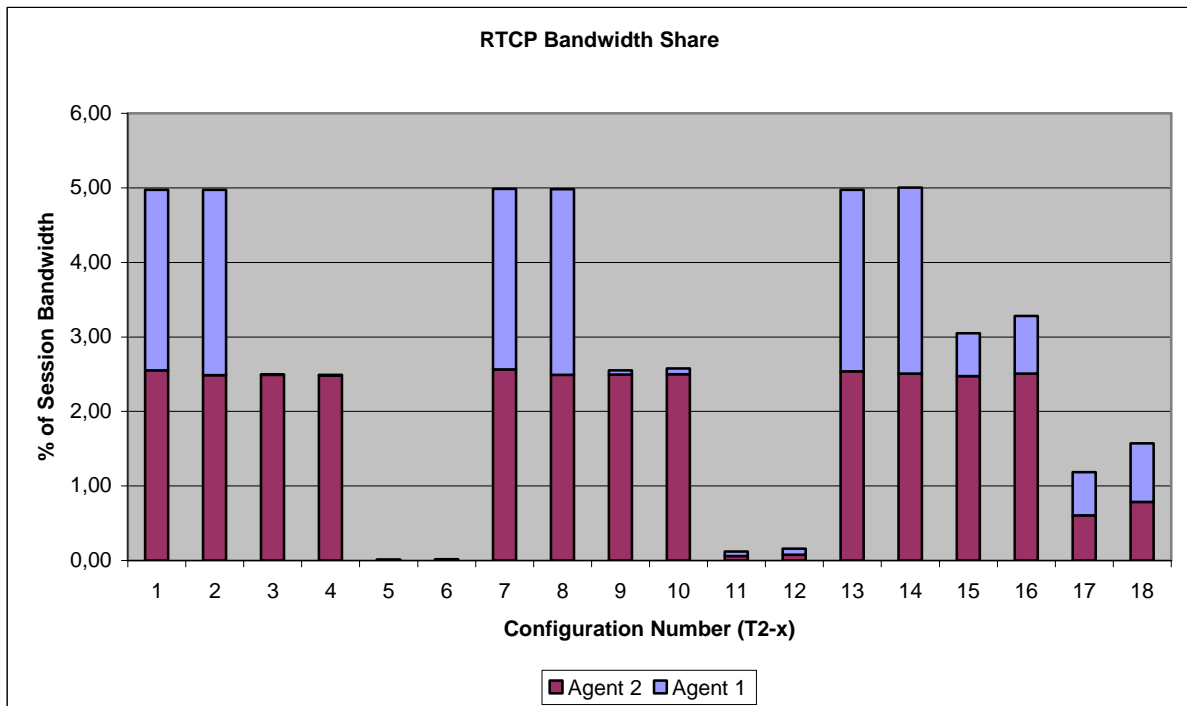


Figure 5: RTCP bandwidth share for the loss less unicast simulations.

#### 4.1.2 Packet Losses with a Constant Loss Rate

In this section we show that the allowed RTCP bandwidth share is not exceeded, even if packet loss occurs. We simulated a constant byte error rate (BYER) on the link. The byte errors are inserted randomly with a uniform distribution. Packets with byte errors are discarded on the link, hence the receiving agents will not see the loss immediately. The agents detect packet loss by a gap in the sequence number.

When the agents detect a packet loss, they feel the need to send feedback. In unicast  $T\_dither\_max$  is always zero, hence an early packet can be sent immediately if  $allow\_early$  is true. If the last packet was already an early one (i.e.  $allow\_early = false$ ), the feedback might be appended to the next regularly scheduled receiver report. The  $max\_feedback\_delay$  parameter (which we set to 1 second in our simulations) determines if that is allowed.

From Figure 6 we see that there is no difference in the RTCP bandwidth share, whether losses occur or not. This is what we expected, because even though the RTCP packet size grows and early packets are sent, the interval between the packets increases and thus the RTCP bandwidth stays the same. Only the RTCP bandwidth of the Agents that use the AVP increases slightly. This is because the interval between the packets is still 5 seconds, but the packet size increased because of the feedback that is appended.

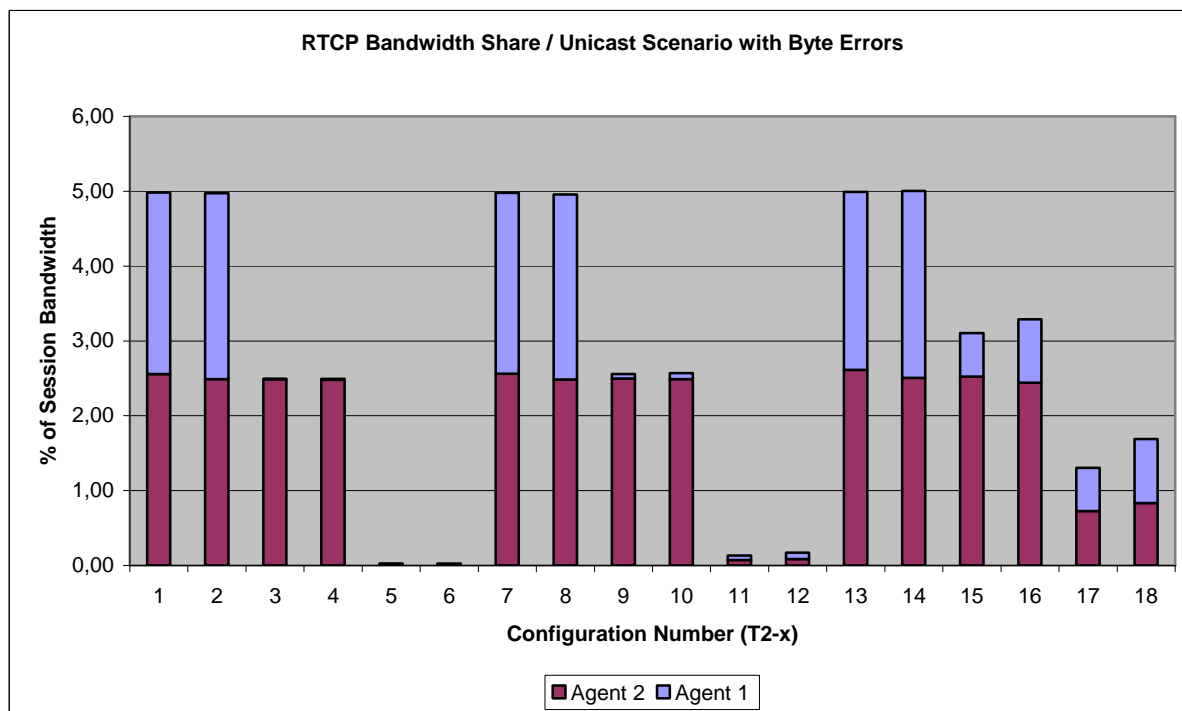


Figure 6: RTCP bandwidth share for unicast scenarios with packet losses

## 4.2 Multicast Topologies

Next we investigated the RTCP bandwidth share in multicast scenarios, i.e. we simulated the topologies T-4, T-8 and T-16 and measured the fraction of the session bandwidth that was used for RTCP packets. Again we considered different situations and protocol configurations (e.g. with or without bit errors, groups with AVP and/or AVPF agents, etc.). The configurations can be seen from the following tables.

Configuration Number	Session Bandwidth	Agents that are RTP senders	Agents that are RTP receivers	Agents using AVP	Agents using AVPF
T4-1	2 Mbps	1	2,3,4	-	1 - 4
T4-2	2 Mbps	3,4	1,2	-	1 - 4
T4-3	2 Mbps	1	2,3,4	1,3	2,4
T4-4	2 Mbps	3,4	1,2	1,3	2,4
T4-5	2 Mbps	1	2,3,4	1 - 4	-
T4-6	2 Mbps	3,4	1,2	1 - 4	-
T4-7	200 kbps	1	2,3,4	-	1 - 4
T4-8	200 kbps	3,4	1,2	-	1 - 4
T4-9	200 kbps	1	2,3,4	1,3	2,4
T4-10	200 kbps	3,4	1,2	1,3	2,4
T4-11	200 kbps	1	2,3,4	1 - 4	-
T4-12	200 kbps	3,4	1,2	1 - 4	-
T4-13	20 kbps	1	2,3,4	-	1 - 4
T4-14	20 kbps	3,4	1,2	-	1 - 4
T4-15	20 kbps	1	2,3,4	1,3	2,4
T4-16	20 kbps	3,4	1,2	1,3	2,4
T4-17	20 kbps	1	2,3,4	1 - 4	-
T4-18	20 kbps	3,4	1,2	1 - 4	-

Table 2: Configurations of the multicast simulations with the T-4 topology

Configuration Number	Session Bandwidth	Agents that are RTP senders	Agents that are RTP receivers	Agents using AVP	Agents using AVPF
T8-1	2 Mbps	1	2 - 8	-	1 - 8
T8-2	2 Mbps	2,6,7	1,3,4,5,8	-	1 - 8
T8-3	2 Mbps	1	2 - 8	1,2,6,8	3,4,5,7
T8-4	2 Mbps	2,6,7	1,3,4,5,8	1,2,6,8	3,4,5,7
T8-5	2 Mbps	1	2 - 8	1 - 8	-
T8-6	2 Mbps	2,6,7	1,3,4,5,8	1 - 8	-
T8-7	200 kbps	1	2 - 8	-	1 - 8
T8-8	200 kbps	2,6,7	1,3,4,5,8	-	1 - 8
T8-9	200 kbps	1	2 - 8	1,2,6,8	3,4,5,7
T8-10	200 kbps	2,6,7	1,3,4,5,8	1,2,6,8	3,4,5,7
T8-11	200 kbps	1	2 - 8	1 - 8	-
T8-12	200 kbps	2,6,7	1,3,4,5,8	1 - 8	-
T8-13	20 kbps	1	2 - 8	-	1 - 8
T8-14	20 kbps	2,6,7	1,3,4,5,8	-	1 - 8
T8-15	20 kbps	1	2 - 8	1,2,6,8	3,4,5,7
T8-16	20 kbps	2,6,7	1,3,4,5,8	1,2,6,8	3,4,5,7
T8-17	20 kbps	1	2 - 8	1 - 8	-
T8-18	20 kbps	2,6,7	1,3,4,5,8	1 - 8	-

**Table 3: Configurations of the multicast simulations with the T-8 topology**

Configuration Number	Session Bandwidth	Agents that are RTP senders	Agents that are RTP receivers	Agents using AVP	Agents using AVPF
T16-1	2 Mbps	1	2 - 16	-	1 - 16
T16-2	2 Mbps	8 - 11	1 - 7, 12 - 16	-	1 - 16
T16-3	2 Mbps	1	2 - 16	3 - 9	1,2,10 - 16
T16-4	2 Mbps	8 - 11	1 - 7, 12 - 16	3 - 9	1,2,10 - 16
T16-5	2 Mbps	1	2 - 16	1 - 16	-
T16-6	2 Mbps	8 - 11	1 - 7, 12 - 16	1 - 16	-
T16-7	200 kbps	1	2 - 16	-	1 - 16
T16-8	200 kbps	8 - 11	1 - 7, 12 - 16	-	1 - 16
T16-9	200 kbps	1	2 - 16	3 - 9	1,2,10 - 16
T16-10	200 kbps	8 - 11	1 - 7, 12 - 16	3 - 9	1,2,10 - 16
T16-11	200 kbps	1	2 - 16	1 - 16	-
T16-12	200 kbps	8 - 11	1 - 7, 12 - 16	1 - 16	-
T16-13	20 kbps	1	2 - 16	-	1 - 16
T16-14	20 kbps	8 - 11	1 - 7, 12 - 16	-	1 - 16
T16-15	20 kbps	1	2 - 16	3 - 9	1,2,10 - 16
T16-16	20 kbps	8 - 11	1 - 7, 12 - 16	3 - 9	1,2,10 - 16
T16-17	20 kbps	1	2 - 16	1 - 16	-
T16-18	20 kbps	8 - 11	1 - 7, 12 - 16	1 - 16	-

**Table 4: Configurations of the multicast simulations with the T-16 topology**

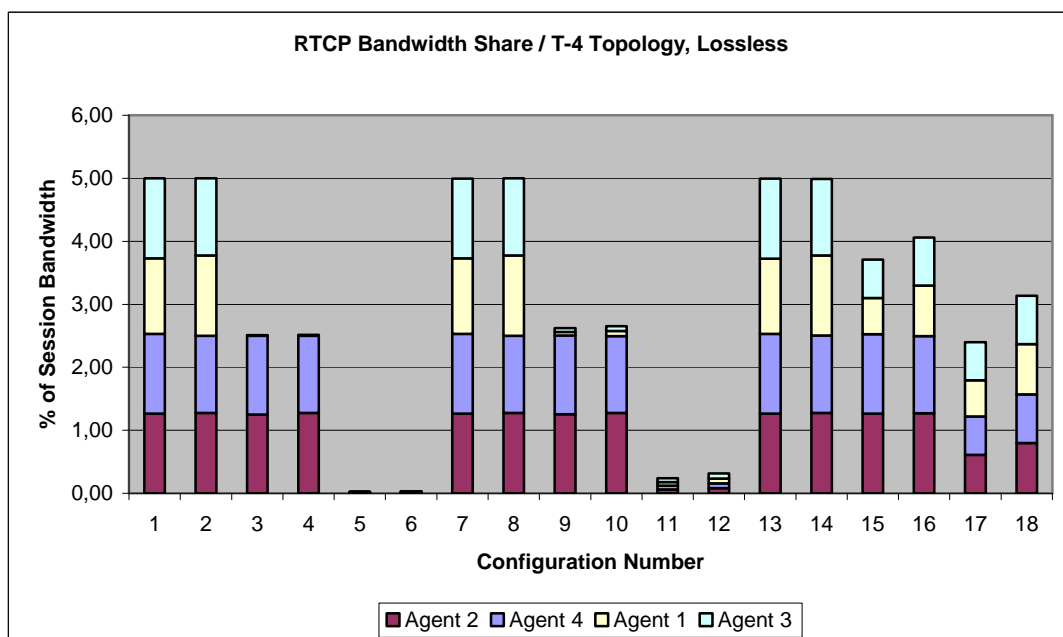
Every configuration should represent a typical use case in the corresponding multicast group size. So for every topology there is a configuration where only one agent is an RTP sender and all the other only receiver, which would be the case for a broadcast or media streaming (i.e. non-interactive) service. The other configurations contain between two and five senders, which represents an interactive media session, e.g. video and audio conferences.

### 4.2.1 No Packet Losses

First we consider the case where no packet loss occurs. As said in the previous chapter, this is also an important study. Even if no early packets are sent the timing rules when to send RTCP feedback are changed, and we want to verify that the allowed RTCP bandwidth share is not exceeded.

The different group sizes are simulated according to the protocol configurations of Table 2, Table 3 and Table 4. We used typical delay settings. The bandwidths of the links is set high enough that no congestion would occur. The results are displayed in the following figures.

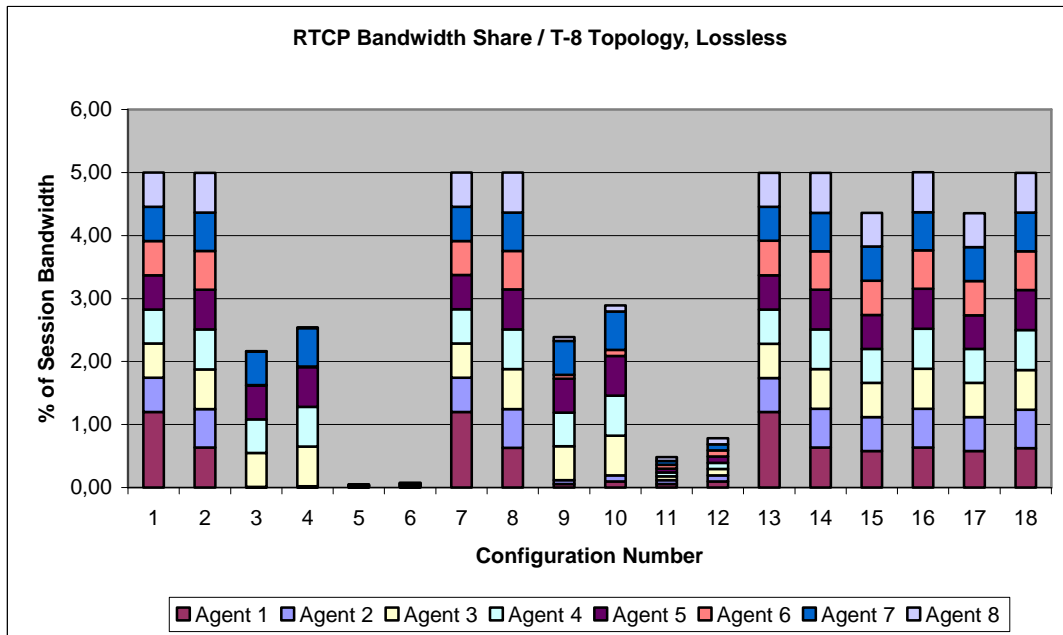
For the small multicast topology T-4 we can see the simulation results in Figure 7. The results are similar to the unicast results in that sense that the RTCP bandwidth is shared fair between the different users and the total amount of 5% of the session bandwidth is not exceeded. In cases of groups where some of the receivers use AVP and some use AVPF, the bandwidth is shared fair according to the rules of AVP and AVPF. The AVPF receiver use the same amount of bandwidth as they would have if the group would consist totally of AVPF receiver. The bandwidth that the AVP receiver do not use, because of the minimum interval of 5s, is not used at all.



**Figure 7: RTCP bandwidth share measured in the multicast simulation of topology T-4 without errors.**

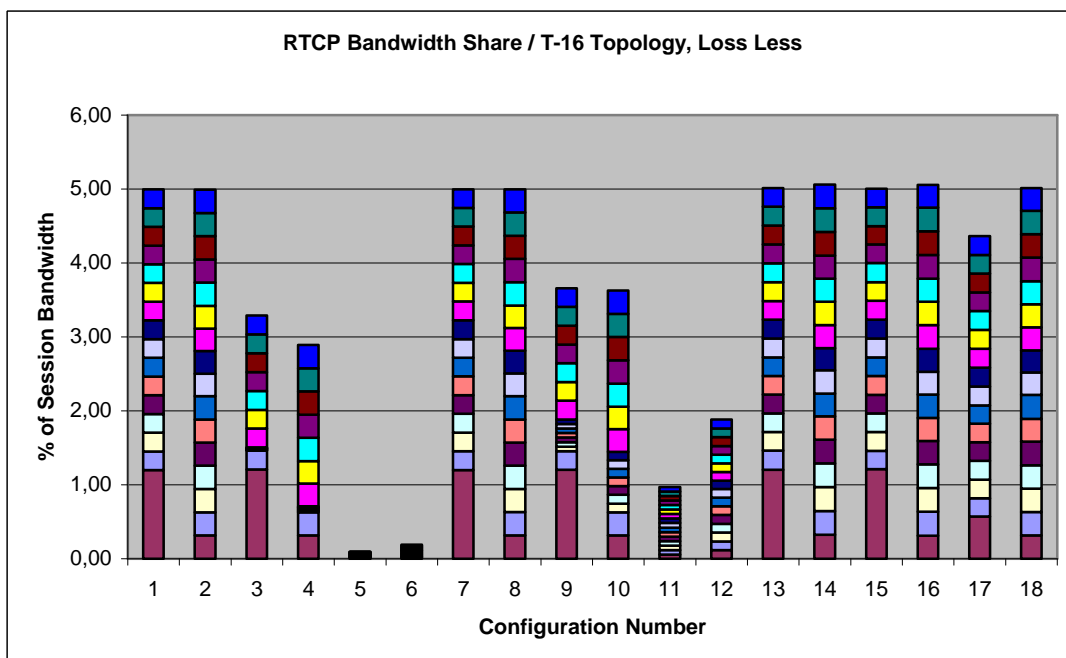
Simulating the larger multicast topology T-8, we get the same results, as can be seen from Figure 8. The group members share the bandwidth fair according to the rules of RTP, AVP and AVPF. In configurations, where only one of the members is a sender (agent 1) and all the others are pure receivers (e.g. configurations 1, 7 or 13), we can see that the sender gets more bandwidth than the receivers. This is due to the RTP rule, that all senders should share at least  $\frac{1}{4}$  of the total RTCP bandwidth. We can see that agent 1 gets more or less exactly  $\frac{1}{4}$  of the RTCP bandwidth (i.e. 1.25% of the total session bandwidth).

In the sessions with a small session bandwidth (i.e. configurations 13-18 with 20kbps session bandwidth), we see that the RTCP bandwidth is already fully used, even if all session members are under restriction of the minimum interval of 5 seconds. Because the available RTCP bandwidth is so low, the calculated  $T_{rr}$  is quite high. But still the receivers using the AVPF have the advantage of being able to send one feedback packet per  $2 \cdot T_{rr}$  more early. The advantages will be explained and shown in greater detail in section ???.



**Figure 8: RTCP bandwidth share measured in the multicast simulation of topology T-8 without errors.**

The simulation of the largest investigated topology with 16 session members, is totally in line with the previous results. Figure 9 shows that the RTCP bandwidth share is not exceeded. The bandwidth is also shared fair among the users. The advantages of using the AVPF are probably lower, especially in case of low session bandwidths. However the the use of AVPF does not lead to any kind of feedback explosion. In larger group sizes the algorithm simply scales to the old rules.



**Figure 9: RTCP bandwidth share measured in the multicast simulation of topology T-16 without errors.**

### 4.2.2 Packet Losses with a Constant Loss Rate

In the following simulations, we introduced a constant packet loss rate on all links. This leads to the receivers wanting to send feedback. The three different multicast topologies are simulated with a high packet loss rate of 1% and a lower one of 0.1%. The packet losses are random with a uniform distribution. The results can be seen from the figures below. We do not show all results for the sake of readability of this documents. Only representative simulation results are given. From Figure 10 and Figure 11 we can see that the RTCP bandwidth share does not differ from scenarios with packet losses to scenarios without. The bandwidth is shared fair among the users as in the loss less cases, described in Section 4.2.1. Also the limit of 5% of the total session bandwidth is not exceeded.

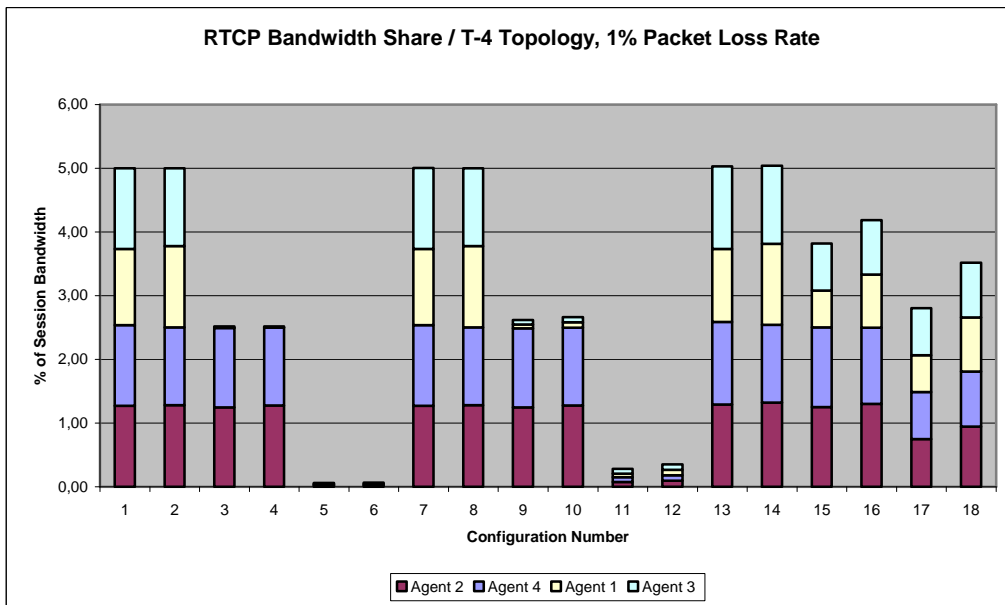


Figure 10: RTCP bandwidth share in the T-4 multicast scenario with random packet losses.

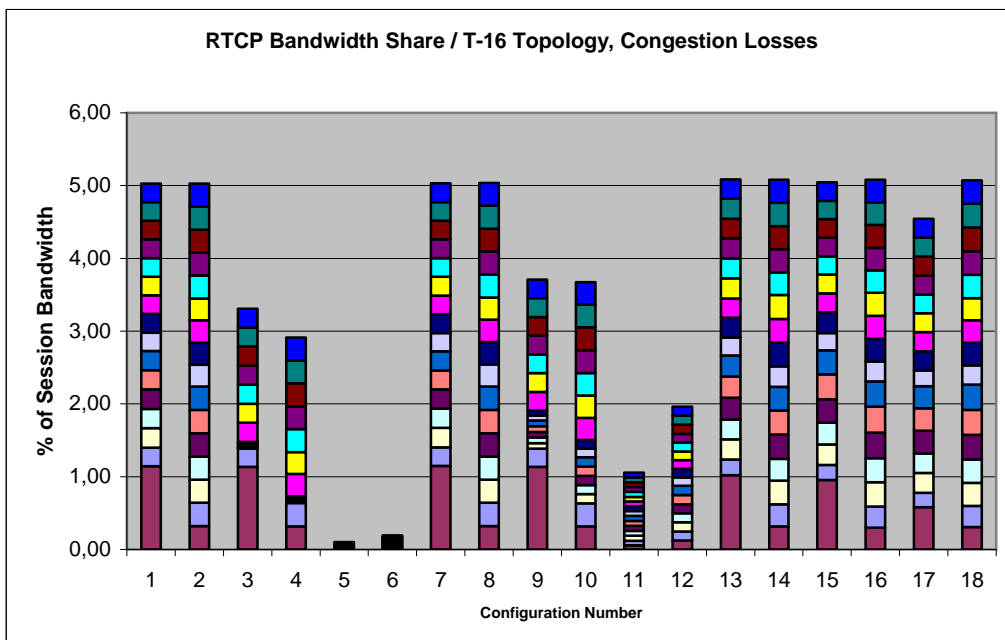


Figure 11: RTCP bandwidth share in the T-16 multicast scenario with congested links and congestion related packet losses.

## 5 Feedback Measurements

In this chapter we describe the results of some feedback delay measurements, we conducted during the simulations. Therefore we use two metrics for measuring the performance of the algorithms, these are the mean “waiting time” (MWT) and the number of feedback that is sent, suppressed or not allowed. The waiting time is the time, measured at a certain agent, between the detection of a packet loss and the time when the corresponding feedback is sent. Assuming that the worth of the feedback decreases with its delay, we think that the mean waiting time is a good metric to measure the performance gain we could get by using AVPF instead of AVP.

The feedback an agent wants to send can be either sent or not sent. If it was not sent, this could be due to the feedback suppression, i.e. another receiver already sent the same feedback or because the feedback was not allowed, i.e. the `max_feedback_delay` was exceeded. We traced for every detected loss, if the agent sent the corresponding feedback or not and if not, why. The more feedback was not allowed, the worse the performance of the algorithm. Together with the waiting times, this gives us a good hint of the overall performance of the scheme.

### 5.1 Unicast Simulations

In the unicast case, the maximum dithering interval `T_Dither_max` is fixed and set to zero. This is due to the fact that it does not make sense for a unicast receiver to wait for other receivers if they have the same feedback to send. But still feedback can be delayed or might not be permitted to be sent at all. The dithering interval is a parameter for the early packets, but at maximum every second packet can be an early packet. The regularly scheduled packets are spaced according to `T_rr`, which depends in the unicast case mainly on the session bandwidth.

Table 5 shows the mean waiting times (MWT) for some configurations of the unicast topology T-2. The number of feedback packets that are sent or why they are not sent is listed also (feedback sent (sent), feedback suppressed (fbs) and feedback not allowed (na)). In some of the cases only agent 1 sends RTP packets and agent 2 is a pure receiver and in others both are sending and receiving. Thus in some cases only agent 2 should send feedback, while in others we measured the statistics for both. To urge the agents to send feedback, we introduce a constant packet loss rate on the link and / or shrink the link bandwidth to be smaller than the session bandwidth, where packets would be lost due to congestion.

From the table above we see that the mean waiting time can be decreased dramatically by using AVPF instead of AVP. While the waiting times for agents using AVP is always around 2.5 seconds (half the minimum interval) it can be decreased to a few ms for most of the AVPF configurations. The waiting times for the bi-directional scenarios are a little smaller, because from one sender only half of the packets are sent. Thus only half of the losses occur. The probability that the receiver is able to send an early packet increases.

In the cases of high session bandwidth normally all feedback is sent. This is because the packet size is quite large (1000byte) and thus per lost packet, more RTCP bandwidth is available. There are only very few exceptions, which are probably due to two packet losses directly after each other and directly after the last regularly scheduled RR was sent. In this case it might be possible that the second feedback could have been sent too late, because  $2 * T_{rr}$  is larger than the `max_feedback_delay`. `T_rr` is calculated with a random factor between 0.5 and 1.5 and thus if by chance the 1.5 is chosen `T_rr` can get quite large.

This is different for the cases with a small session bandwidth. Here we have a small packet size (100byte) and thus many packets are transmitted, while the RTCP bandwidth share is quite low. `T_rr` is thus quite large. After an early packet was sent the time to the next regularly scheduled packet can be very high and be maybe larger than `max_feedback_delay`.

With a different setting of `max_feedback_delay` it is possible to have either more feedback that is not allowed and a decreased mean waiting time or more feedback that is sent but an increased waiting time. Thus the parameter should be set with care according to the application's needs.

Session Bandwidth	Sending Agents	AVPF Agents	Packet Loss Rate	Link Bandwidth	MWT		Feedback Stats Agent 1			Feedback Stats Agent 2		
					A 1	A 2	sent	fbs	n.a.	sent	fbs	n.a.
2 Mbps	1	1,2	0.001	10 Mbps	-	0,015	-	-	-	756	0	0
2 Mbps	1,2	1,2	0.001	10 Mbps	0,011	0,012	376	0	0	377	0	0
2 Mbps	1	1	0.001	10 Mbps	-	2,573	-	-	-	786	0	0
2 Mbps	1,2	1	0.001	10 Mbps	0,010	2,686	410	0	0	410	0	0
2 Mbps	1	-	0.001	10 Mbps	-	2,604	-	-	-	781	0	0
2 Mbps	1,2	-	0.001	10 Mbps	2,675	2,675	395	0	0	387	0	0
2 Mbps	1	1,2	0.01	10 Mbps	-	0,006	-	-	-	7548	0	2
2 Mbps	1,2	1,2	0.01	10 Mbps	0,005	0,004	3747	0	1	3767	0	1
2 Mbps	1	1	0.01	10 Mbps	-	2,551	-	-	-	7363	0	0
2 Mbps	1,2	1	0.01	10 Mbps	0,005	2,557	3796	0	1	3756	0	0
2 Mbps	1	-	0.01	10 Mbps	-	2,591	-	-	-	7480	0	0
2 Mbps	1,2	-	0.01	10 Mbps	2,592	2,657	3780	0	0	3695	0	0
2 Mbps	1	1,2	0	2 Mbps	-	0,001	-	-	-	1741	0	0
2 Mbps	1,2	1,2	0	2 Mbps	0,001	0,001	1755	0	0	1764	0	0
2 Mbps	1	1	0	2 Mbps	-	2,601	-	-	-	1774	0	0
2 Mbps	1,2	1	0	2 Mbps	0,000	2,529	165	0	0	1749	0	0
2 Mbps	1	-	0	2 Mbps	-	2,557	-	-	-	25	0	0
2 Mbps	1,2	-	0	2 Mbps	2,526	2,896	10	0	0	40	0	0
20 kbps	1	1,2	0.001	1 Mbps	-	0,034	-	-	-	74	0	2
20 kbps	1,2	1,2	0.001	1 Mbps	0,000	0,000	31	0	0	36	0	2
20 kbps	1	1	0.001	1 Mbps	-	2,591	-	-	-	80	0	0
20 kbps	1,2	1	0.001	1 Mbps	0,064	2,598	41	0	2	37	0	0
20 kbps	1	-	0.001	1 Mbps	-	2,472	-	-	-	79	0	0
20 kbps	1,2	-	0.001	1 Mbps	2,683	2,645	39	0	0	43	0	0
20 kbps	1	1,2	0.01	1 Mbps	-	0,163	-	-	-	709	0	64
20 kbps	1,2	1,2	0.01	1 Mbps	0,104	0,146	349	0	34	369	0	25
20 kbps	1	1	0.01	1 Mbps	-	2,519	-	-	-	780	0	0
20 kbps	1,2	1	0.01	1 Mbps	0,119	2,639	352	0	31	404	0	0
20 kbps	1	-	0.01	1 Mbps	-	2,605	-	-	-	780	0	0
20 kbps	1,2	-	0.01	1 Mbps	2,537	2,596	398	0	0	364	0	0
20 kbps	1	1,2	0	20 kbps	-	0,162	-	-	-	687	0	70
20 kbps	1,2	1,2	0	20 kbps	0,125	0,136	348	0	29	376	0	37
20 kbps	1	1	0	20 kbps	-	2,540	-	-	-	763	0	0
20 kbps	1,2	1	0	20 kbps	0,089	2,633	332	0	31	404	0	0
20 kbps	1	-	0	20 kbps	-	2,590	-	-	-	780	0	0
20 kbps	1	-	0	20 kbps	2,522	2,611	398	0	0	364	0	0

**Table 5: Mean waiting times, measured in the unicast simulations.**

## 5.2 Multicast Simulations

In this section we describe some measurements of feedback statistics in the multicast simulations. We picked out certain characteristic and representative results. Therefore we considered the topology T-16. Different scenarios and applications are simulated for this topology. The parameters of the different links are set as follows. The agents A2, A3 and A4 are connected to the middle node of the multicast tree, i.e. agent A1, via high bandwidth and low delay links. The other agents are connected to the nodes 2, 3 and 4 via different link characteristics. The agents connected to node 2 represent mobile users. They suffer in certain configurations from a certain byte error rate on their access links and the delays are quite high. The agents that are connected to node 3 have low bandwidth access links, but do not suffer from bit errors. The last agents, that are connected to node 4 have quite high bandwidth and quite low delay. The whole topology and its parameters can be seen in Figure 12.

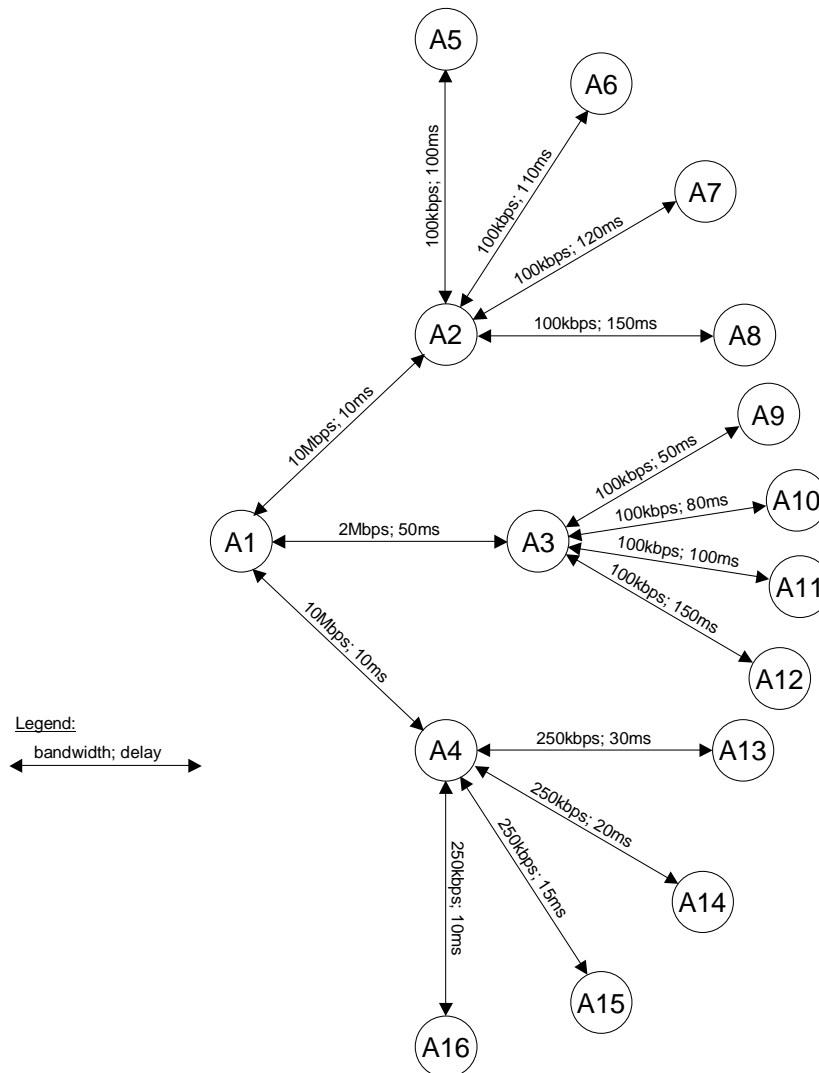


Figure 12: Sample topology for the multicast simulations.

### 5.2.1 Shared Losses vs Distributed Losses

In our first investigation, we wanted to see the influence the loss characteristic on the algorithm's performance, i.e. we wanted to investigate is the case where packet loss occurs for several users simultaneously or totally independently. Therefore we first define agent A1 to be the sender. In the shared-loss-case we insert a constant byte error rate on one of the middle links, i.e. the link between A1 and A2. In the case of distributed losses we inserted the same byte error rate on all links downstream of A2.

This scenario is especially interesting, because of the feedback suppression algorithm. When all receivers share the same loss, it is only necessary for one of them to send the loss report. Hence if a member receives feedback with the same content that it has scheduled to be sent, it suppresses the scheduled feedback. Of course this suppressed feedback does not contribute to the means waiting times. So we expect reduced waiting times for shared losses, because the probability is high that one of the receivers can send the feedback more or less immediately. The results are shown in the following table.

Table 6 shows the feedback statistics for the simulation of a large group size. All 16 agents of topology T-16 joined the RTP session. However only agent A1 acts as an RTP sender, the other agents are pure receivers. Only 4 or 5 agents suffer from packet loss, i.e. A2, A5, A6, A7 and A8 for the case of shared losses and A5, A6, A7

and A8 in the case of distributed losses. Since the number of session members is the same for both cases,  $T_{rr}$  is also the same on the average. Still the mean waiting times are reduced by more than 50% in the case of shared losses. This proves our assumption that shared losses enhance the performance of the algorithm.

The feedback suppression mechanism seems to be working quite fine. Even though some feedback is sent from different receivers (i.e. 1150 loss reports are sent in total and only 650 packets were lost) most of the redundant feedback was suppressed (i.e. 2023 loss reports were suppressed from 3250 individual detected losses).

Agent	Shared Losses					Distributed Losses				
	Feedback				MWT	Feedback				MWT
sent	fbs	na	sum	sent		fbs	na	sum		
A2	274	351	25	650	0,267	-	-	-	-	-
A5	231	408	11	650	0,243	619	2	32	653	0,663
A6	234	407	9	650	0,235	587	2	32	621	0,701
A7	223	414	13	650	0,253	594	6	41	641	0,658
A8	188	443	19	650	0,235	596	1	32	629	0,677

**Table 6: Feedback statistics for multicast simulations.**

## 5.2.2 Sender vs Receiver

RTP senders are able to maintain a RTT measurement to all receivers, which send receiver reports. This is done by the means of the ntp timestamp in the sender report and the repetition of this value together with the delay since last sender report value in the receiver report. However RTP session members that do not send RTP packets are not an RTP sender and thus do not send sender reports. Therefore pure receivers do not have an RTT measurement to the senders or other receivers. This fact is considered in AVPF, by giving two possibilities to calculate  $T_{dither\_max}$ .

If the RTP member has an RTT measurement to the sender of the packet it wants to provide feedback to, it calculates  $T_{dither\_max} = k * T_{rtt}/2 * members$ , with  $k = 1$ . Thus  $t_{dither\_max}$  is increased with the number of session members and the RTT. The rationale for  $RTT/2$  is that the distance to the sender is a good measure how long to wait at maximum. Other receivers, who are more far away, i.e. have a larger RTT estimation, will detect the packets later and also the feedback from those would arrive later and hence have less value. Thus the nearest receivers get the chance first to send their feedback. Because of the larger distance of the other receivers to the sender, they will probably wait longer (probably, because of the randomness, i.e. we calculate  $T_{dither\_max}$ , from which  $T_{dither}$  is picked randomly). While those are waiting, it is likely that they receive the feedback from the receivers that are nearer to the source. With this it is possible to find a good compromise between waiting time and feedback suppression. To let the algorithm scale to large group sizes, the number of session members is included. The number of members is the maximum number of receivers that shared the same loss. The more members are in the session, the higher is the probability that other receivers share the loss and thus the higher is the value of waiting longer, because the probability is increased that feedback suppression will work. If all receivers calculate the same  $T_{dither\_max}$  (i.e. have a similar RTT estimation) and pick a  $T_{dither}$  from this interval randomly with a uniform distribution, it is likely that one feedback is sent within the first RTT interval.

In case the RTP session member does not have an RTT measurement, i.e. it is a pure receiver, it calculates  $T_{dither\_max} = l * T_{rr}$ , with  $l = 1/2$ . The rationale for this is that the receiver, if it has no RTT estimation, does not know at all how long it should wait for other receivers to send feedback. The feedback suppression algorithm would certainly fail, if the time is selected too short. However the waiting time is increased unnecessarily (and thus the value of the feedback is decreased!) in case the time is chosen too long. It would be good to find the optimum time (which is tried to be done with the RTT estimation), but it is not dangerous if the optimum time is not chosen. Decreased feedback value and a failure of the feedback suppression mechanism do not hurt the network stability. We have shown for the cases of distributed losses that the overall bandwidth constraints are kept in any case and thus we could only lose some performance by choosing the wrong time. A good measure for  $T_{dither\_max}$  however is the RTCP interval  $T_{rr}$ . This value increases with the number of session members.

Also we know that we can send feedback at least every  $T_{rr}$ . Thus increasing  $T_{dither\ max}$  beyond  $T_{rr}$  would certainly make no sense. So by choosing  $T_{rr}/2$  we guarantee that at least sometimes (i.e. when a loss is detected in the first half of the interval between two regularly scheduled RTCP packets) we are allowed to send early packets. Because of the randomness of  $T_{dither}$  we still have a good chance to send the early packet in time.

Having said that, we assume that the RTP members who have an RTT measurement would perform better regarding the feedback suppression. We want to show that by simulating the same scenario of Section 5.2.1, but enabling all receivers that suffer from packet loss to maintain a RTT measurement. We do this by declaring the corresponding agents to RTP senders. However we do not send RTP packets from this agents, to be comparable to the previous results. The only difference to the previous simulations is that sender reports are sent, which enables the sender to maintain a RTT measurement.

Table 7 shows the results of the simulations. As assumed, we see that the performance regarding the waiting time is increased significantly. In case of shared losses, the mean time is less than half of the mean waiting times of the receivers that do not have a RTT estimation. Also for the case of distributed losses, we see a slight gain in performance, however not as big as for the shared losses. But still we see that the calculation of  $T_{dither\ max}$ , using the RTT estimation finds a better tradeoff between waiting time and feedback suppression. The waiting time is reduced and the feedback suppression increased where possible. Thus for both cases, whether feedback suppression is possible or not, the performance is increased. Feedback suppression in the case of shared losses is working much better with a RTT estimation. From 3160 individual detected losses only 848 loss reports are sent.

Agent	Shared Losses					Distributed Losses				
	Feedback				MWT	Feedback				MWT
sent	fbs	na	sum	sent		fbs	na	sum		
A2	582	43	7	632	0,100	-	-	-	-	-
A5	70	562	0	632	0,121	644	1	1	646	0,576
A6	60	572	0	632	0,114	638	5	1	644	0,575
A7	73	559	0	632	0,109	607	3	1	611	0,567
A8	63	569	0	632	0,108	626	3	0	629	0,589

**Table 7: Feedback statistics for multicast simulations, where the agents that suffer from packet losses do have a RTT estimation to the sender.**

## 6 Investigations on “k”

The parameter  $k$  in the formula how to calculate  $T\_Dither\_max$  if an RTT estimation is available has some influence of the performance of the algorithm. Thus we investigated the effect and tried to find an optimum value for  $k$ . First we want to investigate the formula for  $T\_Dither\_max$  theoretically. Therefore we calculate the probability that feedback is suppressed under some assumptions. Then we define a sample scenario, where we verify the results. Finally we try to find optimum values for the parameter “ $k$ ”.

### 6.1 Probability of Feedback Suppression

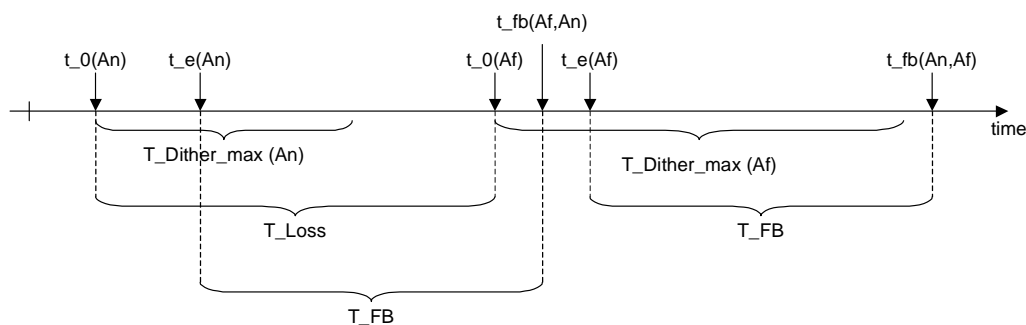
In the following section we use some assumptions to get a quantitative expression for the probability if feedback will be suppressed or not. Therefore we assume the following:

- constant delay between sender and receiving agents
- constant delay between receiving agents
- symmetric links between agents, i.e. same delays for uplink and downlink
- it is always possible to schedule an early packet if a packet loss is detected (i.e. the last feedback sent was a regularly scheduled RR)
- The dithering interval is chosen from the interval between zero and  $T\_Dither\_max$ , i.e. it is not limited by the next regularly scheduled RR

Taking these assumptions into account, Figure 13 illustrates the timeline of a typical loss detection at two agents. Agent  $An$  is quite near to the sender, measured in propagation delay. Agent  $Af$  symbols another agent, which has a larger delay to the sender. Agent  $An$  detects a packet loss at time  $t\_0(An)$  and want to send feedback in an early packet. Therefore it calculates the maximum dithering interval  $T\_Dither\_max(An)$ , which is dependent on its RTT estimation to the sender, the parameter  $k$  and the number of session members,  $members$ . It chooses randomly a point in time from this interval to send the early packet ( $t\_e(An)$ ).

At the time  $t\_0(Af) = t\_0(An) + T\_Loss$ , where  $T\_Loss$  denotes the difference in the propagation delay from the sender to  $An$  and  $Af$ ,  $Af$  will detect the same loss. It will also try to send feedback and thus calculate the dithering interval. However  $T\_Dither\_max(Af)$  will be larger than those of  $An$ , because of the larger RTT estimation to the sender. At the time  $t\_fb(Af,An) = t\_e(An) + T\_FB$  agent  $Af$  will receive the feedback from agent  $An$ .  $T\_FB$  denotes thereby the delay between sending the feedback at agent  $An$  and receiving the same at agent  $Af$ .

At a time  $t\_e(Af)$ , the agent  $Af$  will send his feedback message, if no feedback is received from the other agent before. Thus if  $t\_fb(Af,An) < t\_e(Af)$ , agent  $Af$  will suppress its feedback. Else it will send the feedback which is received from agent  $An$  at the time  $t\_fb(An,Af)$ . If  $t\_fb(An,Af) < t\_e(An)$ , agent  $An$  will suppress its feedback.



**Figure 13: Timing relations for requesting feedback of agents  $An$  and  $Af$  as an example for agents with large RTT differences.**

Taking this information into account we can give expressions, for which the feedback suppression fails: if  $t_{fb}(Af,An) > t_e(Af)$  and  $t_{fb}(An,Af) > t_e(An)$  the feedback is sent at such bad points in time, that the feedback suppression does not work.

The following equations show the intervals from which the points in time are taken randomly with a uniform distribution:

$$\begin{aligned} t_e(An) &\in [t_0(An) ; t_0(An) + T_{Dither\_max}(An)] \\ t_{fb}(Af,An) &\in [t_0(An) + T_{FB} ; t_0(An) + T_{FB} + T_{Dither\_max}(An)] \\ t_e(Af) &\in [t_0(Af) ; t_0(Af) + T_{Dither\_max}(Af)] \\ t_{fb}(An,Af) &\in [t_0(Af) + T_{FB} ; t_0(Af) + T_{FB} + T_{Dither\_max}(Af)] \end{aligned}$$

All these random variables are uniformly distributed and because  $t_e(An)$  and  $t_e(Af)$  are statistically independent, we can calculate the probability that the feedback suppression works, quite easily.

The feedback of Agent  $Af$  will be suppressed if the feedback from agent  $An$  is sent and received before the agent sends its own feedback, i.e.  $t_{fb}(Af,An) < t_e(Af)$ . The probability of this event is given below:

$$P(fb_s) = \begin{cases} 1 & \text{if } T_{FB} + T_{Dither\_max}(An) < T_{Loss} \\ p_1 & \text{if } T_{FB} < T_{Loss} \\ p_2 & \text{if } T_{FB} > T_{Loss} \text{ and } T_{FB} + T_{Dither\_max}(An) < T_{Loss} + T_{Dither\_max}(Af) \\ p_3 & \text{if } T_{FB} > T_{Loss} \text{ and } T_{FB} + T_{Dither\_max}(An) > T_{Loss} + T_{Dither\_max}(Af) \\ 0 & \text{if } T_{FB} > T_{Dither\_max}(Af) + T_{Loss} \end{cases}$$

$$p_1 = \frac{T_{Loss} - T_{FB}}{T_{Dither\_max}(An)}$$

$$\frac{(T_{Dither\_max}(Af) - T_{Dither\_max}(An) - T_{FB}) \cdot (2 \cdot T_{Dither\_max}(Af) + T_{Loss} - T_{FB} - T_{Dither\_max}(Af))}{2 \cdot T_{Dither\_max}(Af) \cdot T_{Dither\_max}(An)}$$

$$p_2 = 1 - \frac{T_{FB} - T_{Loss}}{T_{Dither\_max}(Af)} - \frac{T_{Dither\_max}(An)}{2 \cdot T_{Dither\_max}(Af)}$$

$$p_3 = \frac{(T_{Loss} - T_{FB} + T_{Dither\_max}(Af))^2}{2 \cdot T_{Dither\_max}(Af) \cdot T_{Dither\_max}(An)}$$

Now we calculate the probability that the feedback from agent  $An$  is suppressed, because the feedback from agent  $Af$  was already received, i.e.  $P(t_{fb}(An,Af) < t_e(An))$ :

$$P(fb_s) = \begin{cases} 0 & \text{if } T_{FB} + T_{Loss} > T_{Dither\_max}(An) \\ \frac{(T_{Dither\_max}(An) - T_{Loss} - T_{FB})^2}{2 \cdot T_{Dither\_max}(Af) \cdot T_{Dither\_max}(An)} & \text{else} \end{cases}$$

In the following sections we verify these formulas (if the assumptions are more or less fulfilled) by simulations.

## 6.2 Sample Scenarios

For the verification of the formulas from the previous section, we define three representative sample scenarios. We use the topology from Section 5.2. Most of the agents however contribute only little to the simulations, because we introduce an error rate only on the link between the sender A1 and the agent A2.

The first scenario represents cases, where losses are shared between two agents. One agent is located upstream on the path between the other agent and the sender. Therefore agent A2 and agent A5 see the same losses, that are introduced on the link between the sender and agent A2. Agent A6, A7 and A8 do not join the RTP session. From the other agents only agents A3 and A9 join. Both agent A2 and A5 are declared as RTP senders, in order to have an RTT estimation to the sender A1.

The second scenario represents also cases, where losses are shared between two agents, but this time the agents are located on different branches of the multicast tree. The delays to the sender are roughly of the same magnitude. Agent A5 and A6 share the same losses. Agents A3 and A9 join the RTP session, but are pure receivers and do not see any losses.

Also in the third scenario, the losses are shared between two agents, A5 and A6. The same agents as in the second scenario are active. However the delays of the links are different. The delay of the link between agent A2 and A5 is reduced to 20ms and between A2 and A6 to 40ms. Thus the RTT estimations of agents A5 and A6 to the sender are reduced significantly.

The following tables summarize the parameter settings of the two sample scenarios:

Parameter	Scenario 1	Scenario 2	Scenario 3
delay sender-agent An	10 ms	110 ms	30 ms
delay sender-agent Af	110 ms	120 ms	50 ms
T_Loss	100 ms	10 ms	20 ms
T_FB	100 ms	210 ms	60 ms
members	5	5	5
k	(0.1, ...,4)	(0.1, ...,4)	(0.1, ...,4)

**Table 8: Parameters of the sample scenarios for the “k” investigations.**

## 6.3 Feedback Suppression in the Sample Scenarios

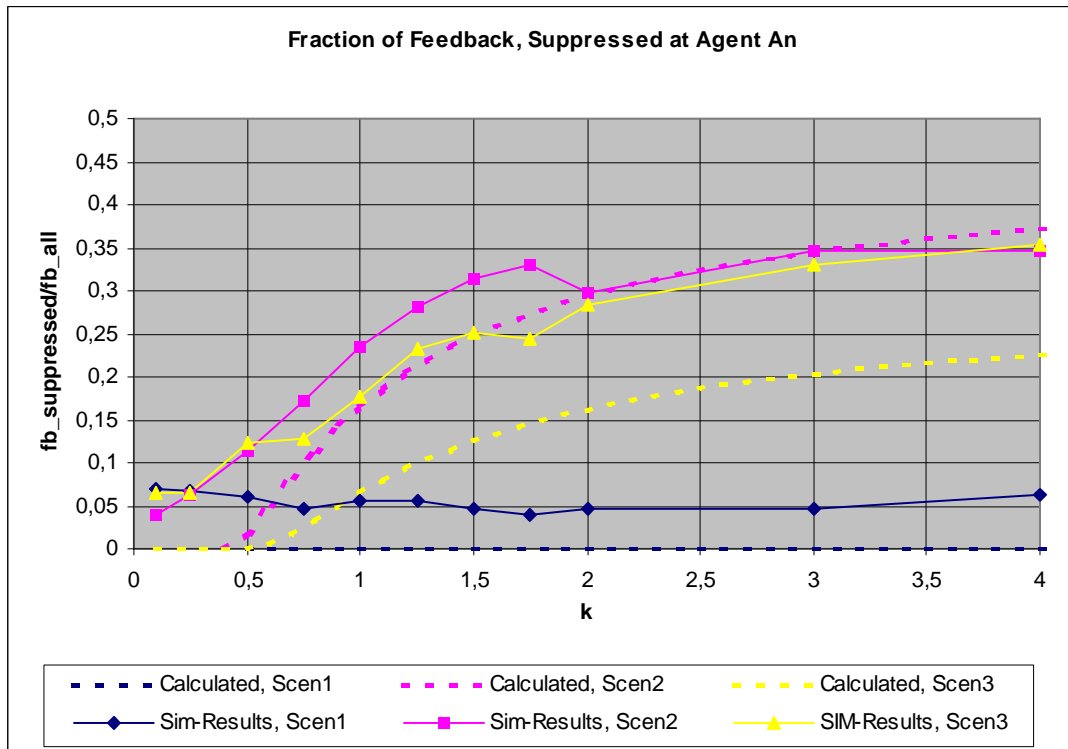
First we calculated for the given parameters of the sample scenarios the probability that the agents suppress the feedback and also that the feedback suppression fails. Therefore we used the formulas that were developed in Section 6.1. Then we simulated the scenarios. The results of the calculated values compared to the simulation results can be found in the following figures.

First we consider the probability that the agent that is nearer to the sender suppresses its feedback, because it received feedback from the farther agent. The calculated results as well as the simulation results can be seen in Figure 14. In general it can be seen that agent An suppresses more feedback if the delay differences are lower. This is only reasonable, because the feedback from other receivers will be faster received in that case. It can also be seen that the feedback suppression rate increases with k. This is due to the fact that  $T_{\text{dither\_max}}$  increases with k. Thus the agents will wait longer on the average before sending their feedback. By increasing the waiting time for all agents, the time where feedback suppression is possible at all is increased.

In the first Scenario, the calculated probability of feedback suppression is nearly zero for all k. This is because, if the feedback is purely determined by  $T_{\text{dither\_max}}$ , the interval from which the feedback sending time is taken at agent An is totally below the time when the feedback can be received from agent Af. However we see that in the simulations, some feedback actually is suppressed. This can be explained, by the fact that not all assumptions, that we made for the calculations are valid in all cases. The feedback time is not only determined by the loss event and the random value from the interval from zero to  $T_{\text{dither\_max}}$ . If  $t_0 + T_{\text{dither\_max}}$  is larger than the

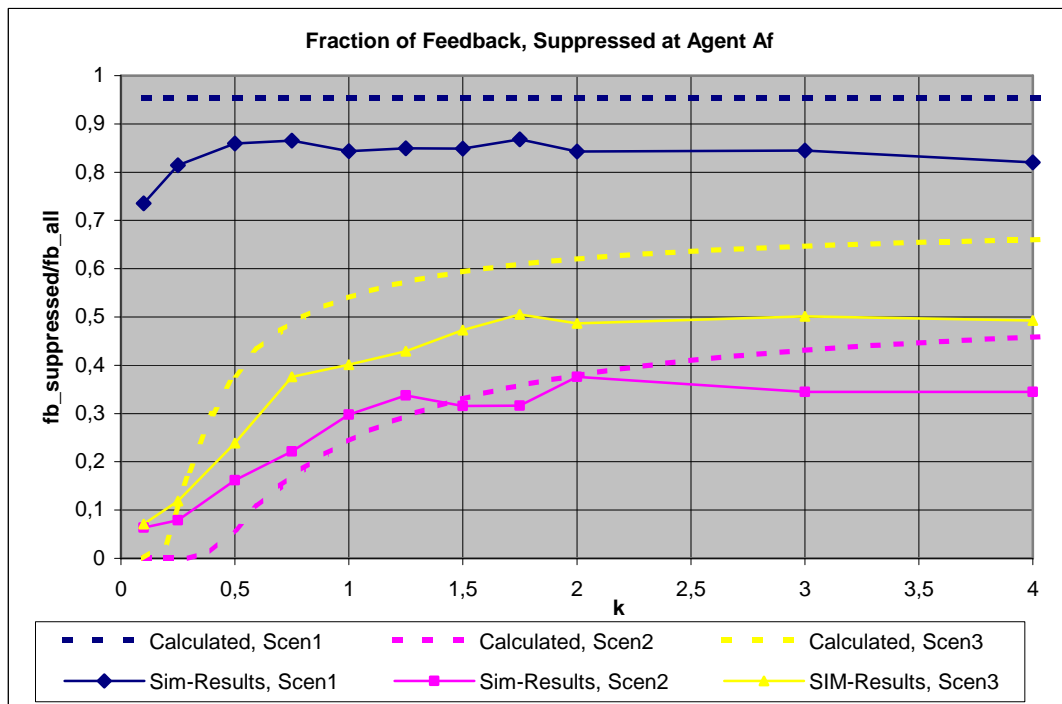
time of the next regularly scheduled receiver report, no early packet would be sent. Instead the feedback is appended to the next RR. Due to timer reconsideration, the scheduled time can be delayed, maybe even for a significant amount of time. In bad cases the scheduled RR can be delayed for one period of  $T_{rr}$ . Another reason for the suppressed packets can be that the agent was not always capable of sending an early packet. It might be that directly after a regularly scheduled RR, a loss was detected. Thus the agent schedules an early packet and by chance is able to send it quite soon. If the next loss it detected, the agent is not allowed to send an early packet, but instead has to append the feedback to the next regularly scheduled RR, which might be sent in nearly two times  $T_{rr}$  time.

The same applies for scenario 2 and 3. Because of the longer waiting times in the cases, where the assumptions are broken, more feedback can be suppressed from agent An. However we see that the trend that we calculated is reflected by our simulations.



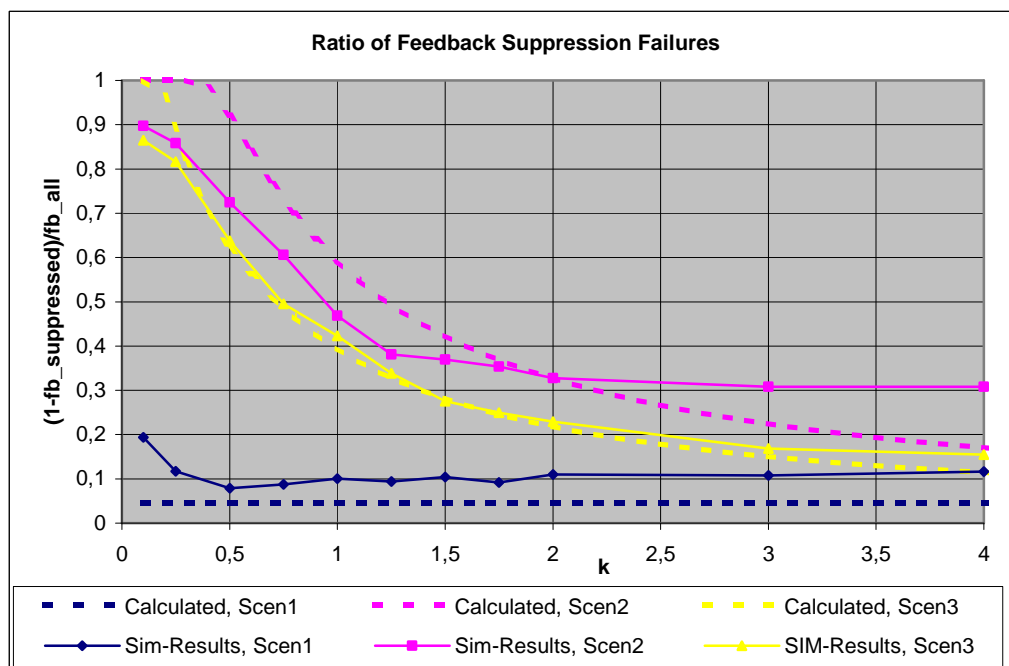
**Figure 14: Fraction of feedback that was suppressed at agent An of the total number of feedback the agent wanted to send.**

In Figure 15 the results for the feedback suppression of agent Af are depicted. Again we see that the number of feedback suppressions increase with  $k$ . Only in scenario 1 the number is more or less constant. We could see that already for the formula, which depends on the term  $T_{Dither\_max}(An)/T_{Dither\_max}(Af)$  only, in the case that the feedback delay is the same as the loss delay difference. Since  $T_{Dither\_max}$  is proportional to  $k$ , the feedback suppression rate should be independent from  $k$ . This can be seen in the calculated graph of scenario 1. The simulation results show a slightly different graph: the feedback suppression rate is smaller. As described above, here also some assumptions are not valid in some cases. Again the waiting time can be increased by the reasons given above. However by increasing the waiting times, the probability that the feedback is suppressed is decreased at agent Af.  $k=2$  seems to be the threshold, where the feedback suppression does not change anymore in the given scenarios. This is because for the given parameters, the early packets will not be sent any more, because the next regularly scheduled RTCP packet will be within the  $T_{dither\_max}$  interval.



**Figure 15: Fraction of feedback that was suppressed at agent Af of the total number of feedback the agent wanted to send.**

In Figure 16 the ration of feedback suppression failures is illustrated. In general the observations from the figures above are summarized. The ratio of feedback failures decreases with an increasing k for the scenarios 2 and 3. In scenario 1 the ratio is hardly influenced at all by k. The simulations show a kind of steady state at k larger 2 or three, where the rational for this is that for very large k, T\_dither\_max becomes equal or more than T\_rr and thus no early packets are send any more. The maximum dithering interval is for these cases limited by the next regularly scheduled RR.



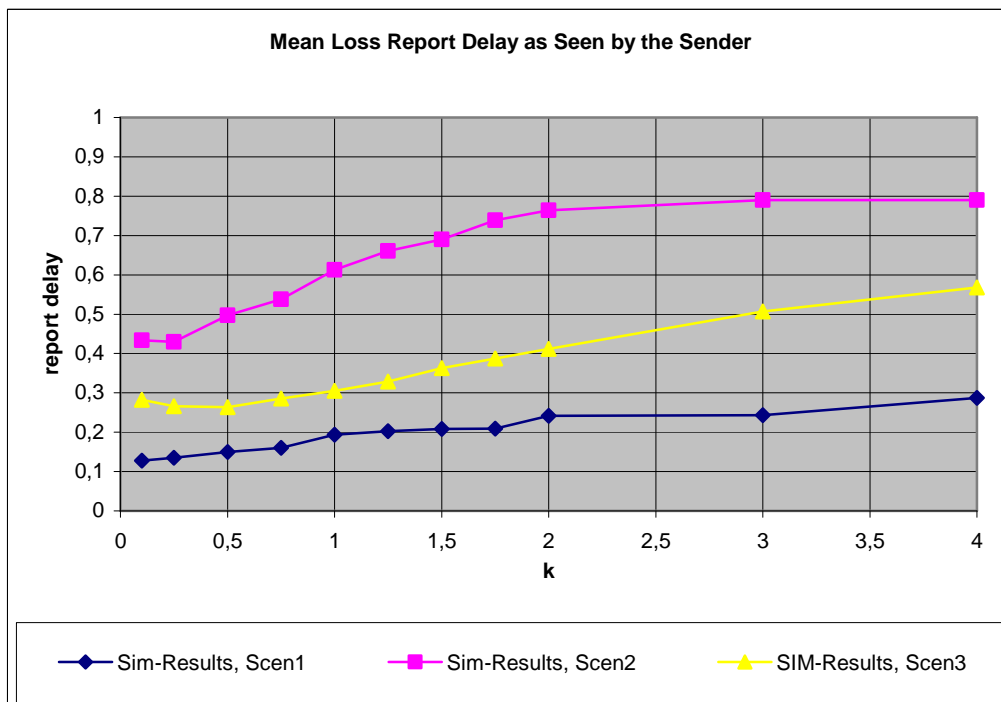
**Figure 16: The ratio of feedback suppression failures.**

Summarizing it can be said, that the theoretical formulas to calculate the probability of feedback suppression are valid under certain circumstances. Under normal conditions, they still represent a rough tendency. The feedback suppression performance is highly dependent on the topology, the parameters and configurations. In general a larger value for  $k$  increases the probability that the feedback suppression works, however the performance gain decreases with an increasing  $k$ . For a certain threshold, depending on the configuration and environment, an increasing  $k$  does not lead to any performance gain any more.

## 6.4 Loss Report Delay

In this section we investigate the influence of the parameter  $k$  on the loss report delay. Therefore we measured for the three sample scenarios the mean loss report delay as seen by the sender, i.e. the sender calculates for every loss report, it receives for the first time the delay since the corresponding packet was sent.

The results are depicted in Figure 17. In general it can be said, that the loss report delay increases with  $k$ . This is only natural, because  $T\_Dither\_max$  is proportional to  $k$ . Thus the agents wait on the average longer to send their early packets. In cases of very large  $k$  values, the report delay does not increase significantly any more. In these cases nearly no early packets are sent, because the next regularly scheduled packet is within the  $T\_dither\_max$  interval. The threshold of  $k$ , from which on the delay will not increase, is dependent on the RTT estimation. For increasing RTT values, the threshold decreases. We see that in scenario 1 the threshold lies between  $k=2$  and  $k=3$ . For the scenarios with smaller RTT, the threshold is higher.



**Figure 17: The mean loss report delay for the three sample scenarios, as measured by the sender.**

Summarizing it can be said, that the report delay increases with an increasing  $k$ . From a certain threshold the increase is not significant, however this threshold is highly dependent on topology and environment parameters.

## 6.5 Summary of “k”-Investigations

We have shown, theoretically as well as by simulations, that the parameter  $k$  influence the feedback performance. While in general the feedback suppression performance increases with  $k$ , the report delay increases also. Hence we need to find a tradeoff, between the amount of feedback that is sent and the delay of the feedback, when it is

received at the sender. Since we have shown that the performance curves for the feedback suppression as well as the report delay is highly variable for different topologies and environments, it is not possible to give an optimized parameter value for  $k$ . We think that  $k=1$  is a compromise, which should be acceptable for most of our considered cases. At least we guarantee with  $k=1$  that no feedback explosion will occur and thus keep the network stability untouched.

## 7 Investigations on “l”

In this section we want to investigate the influence of the parameter “l” from the  $T\_Dither\_max$  calculation in agents that do not have an RTT estimation to the sender. As we have done in the previous section for the parameter “k” we try to evaluate the feedback suppression probability first theoretically under certain assumptions. Then we verify the calculations by the means of simulating the feedback suppression probability for three sample scenarios. Third we show the report delay that results from our simulations of the sample scenarios.

### 7.1 Probability of Feedback Suppression

In general the same theory as in Section 6.1 applies also for the case where the agents do not have a RTT estimation. The only difference is the calculation of  $T\_Dither\_max$ , where all agents calculate the same value, depending only on  $T\_rr$ . Thus under the same assumptions as in Section 6.1, the following formula applies for the probability that the feedback from agent  $A_f$  is suppressed:

$$P(fb_s) = \begin{cases} 1 & \text{if } T\_FB + T\_Dither\_max < T\_Loss \\ p_1 & \text{if } T\_FB < T\_Loss < T\_FB + T\_Dither\_max \\ p_2 & \text{if } T\_FB > T\_Loss \\ 0 & \text{if } T\_FB > T\_Dither\_max + T\_Loss \end{cases}$$

$$p_1 = \frac{T\_Loss - T\_FB}{T\_Dither\_max} + \frac{T\_FB \cdot (T\_Dither\_max + T\_Loss - T\_FB)}{2 \cdot T\_Dither\_max^2}$$

$$p_2 = \frac{(T\_Loss - T\_FB + T\_Dither\_max)^2}{2 \cdot T\_Dither\_max^2}$$

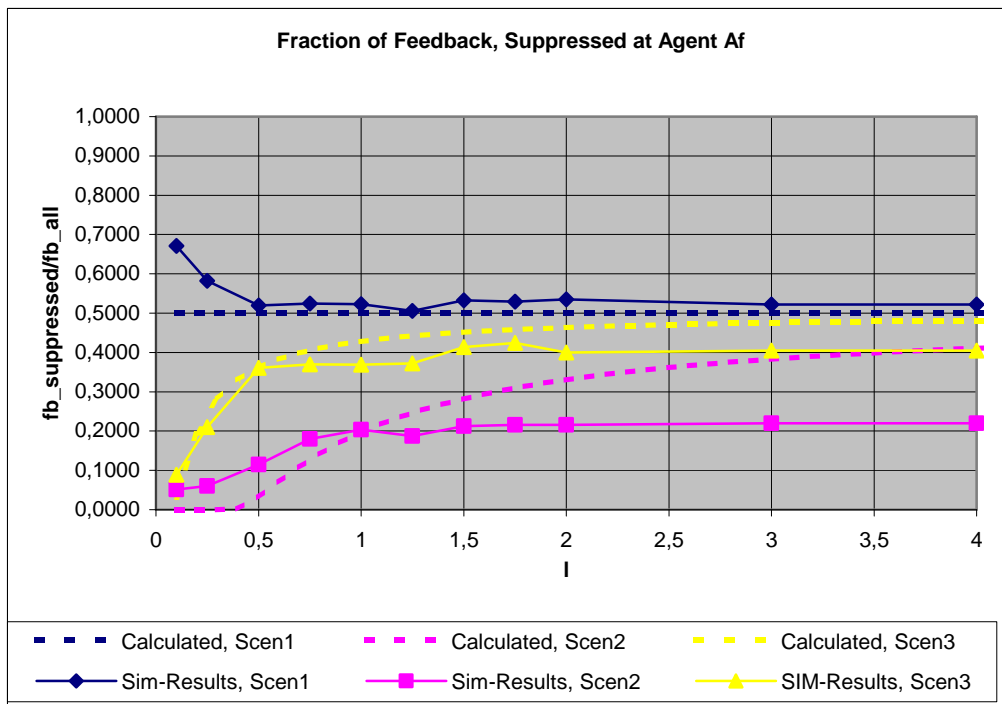
Similar as in Section 6.1 we calculate now the probability that the feedback from agent  $A_n$  is suppressed, because the feedback from agent  $A_f$  was already received, i.e.  $P(t\_fb(A_n, A_f) < t\_e(A_n))$ :

$$P(fb_s) = \begin{cases} 0 & \text{if } T\_FB + T\_Loss > T\_Dither\_max \\ \frac{(T\_Dither\_max - T\_Loss - T\_FB)^2}{2 \cdot T\_Dither\_max \cdot T\_Dither\_max} & \text{else} \end{cases}$$

### 7.2 Feedback Suppression in the Sample Scenarios

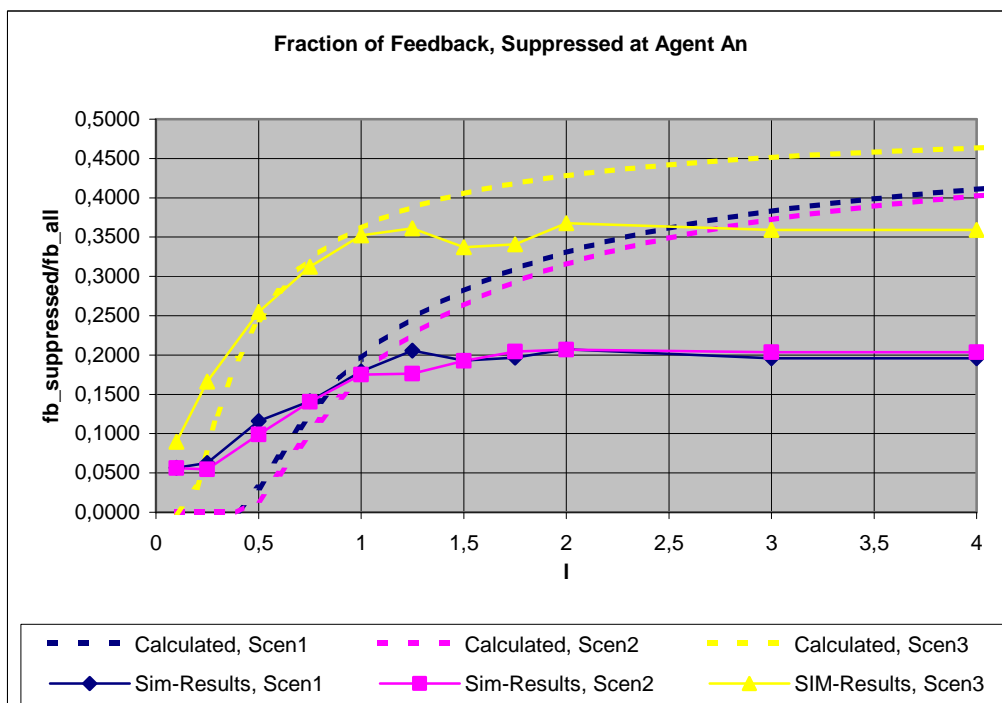
We simulated the three sample scenarios as described in Section 6.2, but this time the all agents beside agent  $A_1$  are pure RTP receivers. Thus these agents do not have an RTT estimation to the source.  $T\_Dither\_Max$  is calculated with the other formula, depending only on  $T\_rr$  and  $l$ , which means that all agents should calculate roughly the same  $T\_Dither\_Max$ .

The results for the feedback suppression rate of the agent  $A_f$  that is more far away from the sender, are depicted in. First of all, we see that the theoretical results are similar to the simulated ones, especially in ranges where  $l$  is not too small or too large. For large  $l$  the same as for the  $k$  investigations apply: the assumptions are not fulfilled anymore, i.e. feedback is mainly sent in regularly scheduled RR and nearly no early packets are sent at all. In general it can be seen that the feedback suppression rate increases with an increasing  $l$ . However there is a threshold, depending on the environment, from which the additional gain is not significant any more.



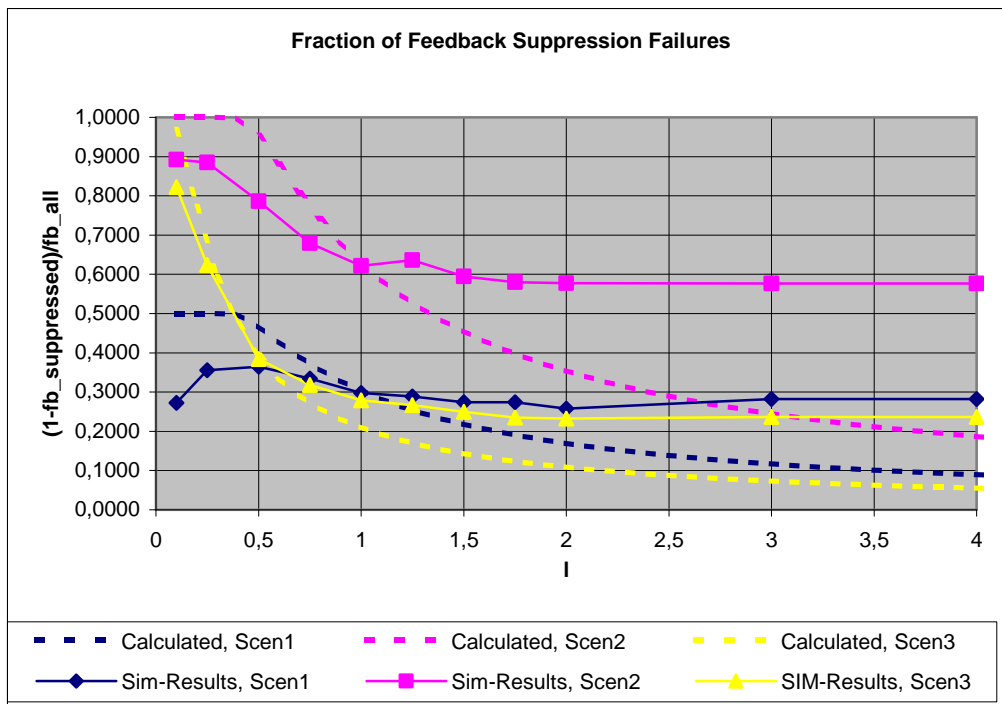
**Figure 18: Feedback suppression rate at the far agent.**

Similar results can be seen for the agent that is nearer to the sender in ???. However the effect that the theoretical curves are different from the simulation results for large values of l is increased. This is because the reduced dithering interval (because of the next regularly scheduled RR), which we did not take into account for our calculations, has greater influence for the near agent. In general the results are the same as for the far agent.



**Figure 19: Feedback suppression rate at the near agent.**

The rate of feedback suppression failure is depicted in ???. The trend that the additional performance increase is not significant from a certain threshold, depending on the environment is here as well visible.



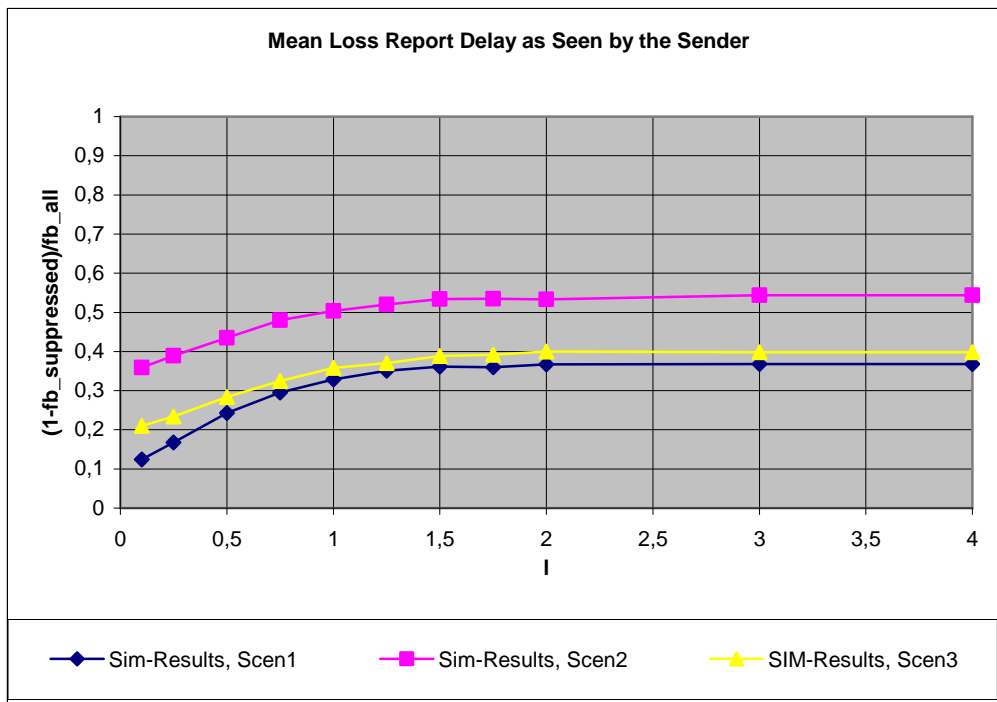
**Figure 20: Total rate of feedback suppression failures.**

Summarizing the feedback suppression results it can be said that in general the feedback suppression performance increases with an increasing  $l$ . However from a certain threshold, depending on environment parameters such as propagation delays or session bandwidth, the additional increase is not significant anymore.

### 7.3 Loss Report Delay

In this section we show the results for the measured report delay during the simulations of the three sample scenarios. This measurement is a metric of the performance of the algorithms, because the value of the feedback for the sender typically decreases with the delay of its reception. The loss report delay is measured as the time at the sender between sending a packet and receiving the first corresponding loss report.

As can be seen from Figure ??? the delay increases in general with an increasing value of  $l$ . However a similar effect as for the feedback suppression performance is visible: from a certain threshold, the additional increase in delay is not significant anymore. The threshold is environment dependent and seems to be related to the threshold, where the feedback suppression gain would not increase anymore.



**Figure 21: Mean loss report delay at the sender.**

## 7.4 Summary of “l”-Investigations

We have shown that theoretically the performance of the feedback suppression mechanisms is increasing with an increasing value of  $l$ . This was also shown practically by simulations. The same applies for the report delay, which increases also with an increasing  $l$ . However the simulations show that in ranges of large  $l$  values the theory is not valid, because assumptions are not fulfilled. This leads to a threshold where both the performance and the delay does not increase any further. The threshold is environment dependent.

So finding an optimum value of  $l$  is not possible because it is always a tradeoff between delay and feedback suppression performance. With  $l=0.5$  we think that a tradeoff was found that is acceptable for typical applications and environments.

## 8 Summary

The new RTP profile AVPF was investigated regarding performance and potential dangers to the network stability. Simulations were conducted using the network simulator, simulating unicast and different sized multicast topologies. The results were shown in this document.

Regarding the network stability, it was important to show that the new profile does not lead to any feedback explosion, or use more bandwidth as it is allowed. Thus we measured the bandwidth that was used for RTCP in relation to the RTP session bandwidth. We have shown that more or less exactly 5% of the session bandwidth is used for RTCP, in all considered scenarios. The scenarios included unicast with and without bit errors, different sized multicast groups, with and without errors or congestion on the links. Thus we can say that the new profile behaves network friendly in that sense that it uses only the allowed bandwidth that was assigned by RTP.

Second we have shown that receivers using the new profile experience a performance gain. We have shown that especially RTP receiver that do have an RTT estimation to the sender gain from using the new profile. But also the other receivers could increase their performance. This was measured by the delay that the sender sees for the received feedback. Using the new profile this delay can be decreased by orders of magnitude.

Third we investigated certain parameters of the new algorithms. We have shown that there does not exist an optimum value for those. The influence of the parameters is highly environment specific and a tradeoff between performance of the feedback suppression algorithm and the experienced delay has to be found. The values that are given in the draft seem to be reasonable for most applications and environments.

## Bibliography

- [AVPF] J.Ott, S.Wenger, S.Fukunaga, N.Sato, K.Yano, A.Miyazaki, K.Hata, R.Hakenberg, C.Burmeister: Extended RTP Profile for RTCP-based Feedback, Internet Draft, draft-ietf-avt-rtcp-feedback-00.txt, Work in Progress, July 2001.
- [RTP] H.Schulzrinne, S.Casner, R.Frederick, and V.Jacobson: RTP - A Transport Protocol for Real-time Applications, Internet Draft, draft-ietf-avt-rtp-new-10.txt, Work in Progress, July 2001.
- [AVP] H.Schulzrinne, S.Casner: RTP Profile for Audio and Video Conferences with Minimal Control, Internet Draft, draft-ietf-avt-profile-new-11.txt, Work in Progress, July 2001.

## Annex A: Probability Functions

Given two intervals  $[a_1, a_2]$  and  $[b_1, b_2]$  and two variables  $a$  and  $b$ , which are drawn independently randomly with a uniform distribution, we calculate the probability that  $a < b$ .

The random variables are uniformly distributed, which results in a distribution function of:

$$f_a(x) = \begin{cases} \frac{1}{a_2 - a_1} & \text{if } a_1 < x < a_2 \\ 0 & \text{else} \end{cases}$$

$$f_b(x) = \begin{cases} \frac{1}{b_2 - b_1} & \text{if } b_1 < x < b_2 \\ 0 & \text{else} \end{cases}$$

The probability that  $a < b$  can be expressed as follows:

$$P(a < b) = \int_x P(x < a < x + dx | b > x)$$

We consider the following cases:

- i.  $a_1 < b_1$  and  $a_2 < b_2$
- ii.  $a_1 < b_1$  and  $a_2 > b_2$
- iii.  $a_1 > b_1$  and  $a_2 < b_2$
- iv.  $a_1 > b_1$  and  $a_2 > b_2$

Case i.:  $a_1 < b_1$  and  $a_2 < b_2$

$$\begin{aligned} P(a < b) &= \int_x P(x < a < x + dx | b > x) \\ &= \int_{a_1}^{b_1} \frac{1}{a_2 - a_1} dx + \int_{b_1}^{a_2} \frac{(b_2 - x)}{(b_2 - b_1)(a_2 - a_1)} dx \\ &= \frac{b_1 - a_1}{a_2 - a_1} + \frac{2b_2(a_2 - b_1) - (a_2^2 - b_1^2)}{2(b_2 - b_1)(a_2 - a_1)} \end{aligned}$$

Case ii.:  $a_1 < b_1$  and  $a_2 > b_2$

$$\begin{aligned} P(a < b) &= \int_x P(x < a < x + dx | b > x) \\ &= \int_{a_1}^{b_1} \frac{1}{a_2 - a_1} dx + \int_{b_1}^{b_2} \frac{(b_2 - x)}{(b_2 - b_1) \cdot (a_2 - a_1)} dx \\ &= \frac{b_1 - a_1}{a_2 - a_1} + \frac{(b_2 - b_1)^2}{2(b_2 - b_1)(a_2 - a_1)} \end{aligned}$$

Case iii.:  $a_1 > b_1$  and  $a_2 < b_2$

$$\begin{aligned}
 P(a < b) &= \int_x P(x < a < x + dx \mid b > x) \\
 &= \int_{a_1}^{a_2} \frac{(b_2 - x)}{(b_2 - b_1) \cdot (a_2 - a_1)} dx \\
 &= \frac{b_2 - \frac{(a_2 + a_1)}{2}}{(b_2 - b_1)}
 \end{aligned}$$

Case iv.:  $a_1 > b_1$  and  $a_2 > b_2$

$$\begin{aligned}
 P(a < b) &= \int_x P(x < a < x + dx \mid b > x) \\
 &= \int_{a_1}^{b_2} \frac{(b_2 - x)}{(b_2 - b_1) \cdot (a_2 - a_1)} dx \\
 &= \frac{(b_2 - a_1)^2}{2(b_2 - b_1)(a_2 - a_1)}
 \end{aligned}$$