Fast RTP Retransmission for IPTV - Implementation and Evaluation

M.J. Prins, M. Brunner, G. Karagiannis, H. Lundqvist, and G. Nunzi

Abstract—**With the deployment of IPTV reliability for multicast is becoming an important research topic again. Even though it has been intensively investigated before, there is now an understanding of the deployment scenario and the application requirements that allows solutions to be evaluated in detail. We describe how to design a fast retransmission cache based on recent extensions of RTP. We have implemented a prototype intended for deployment in an access node and explain the necessary trade-offs in the design. The paper also contains a performance evaluation which shows the efficiency of the retransmission functionality to handle losses and its performance in congested networks.**

Index Terms—**IPTV, QOS, ARQ, RTP.**

I. Introduction

A low packet loss rate is essential to provide good quality for IPTV services. Because of the error propagation properties of interframe video coding a single lost packet can cause quality degradations that can last in the order of a second depending on the encoding parameters. Even if an operator uses quality of service mechanisms that make losses unlikely within its own network, losses may occur in home networks, which are usually not quality of service aware. It is beneficial for the operator to handle also these losses to maximize the customer satisfaction and make the customers more inclined to buy IPTV services. A customer that experiences bad quality due to problems in the home network is not likely to buy IPTV services.

For IPTV services sent by multicast an end-to-end retransmission scheme would lead to feedback implosion when receivers notify the source about what packets they need to get retransmissions of. An alternative solution is to use forward error correction (FEC) end-to-end so that the receivers can recover a certain amount of losses [1]. However, in a multicast solution the amount of added parity information is identical for all users, which means that when the loss rates vary for the different receivers there will either be some users with remaining losses or bandwidth will be wasted in large parts of the network where the loss rate is low.

M. Brunner, H. Lundqvist and G. Nunzi are with NEC Laboratories Europe, Heidelberg, Germany (e-mail: {brunner, lundqvist, nunzi@nw.neclab.eu}).

M.J.Prins is with University of Twente, Enschede, Netherlands (e-mail: m.j.prins-1@alumnus.utwente.nl).

G. Karagiannis is with CTIT - University of Twente, Enschede, Netherlands (e-mail: g.karagiannis@utwente.nl).

Another solution is to use local loss recovery for smaller parts of the multicast tree. By introducing a fast retransmission function in an access node, losses can be recovered rapidly and the quality for the users can be maintained. The proposed retransmission function can use the play-out buffer at the receiver to handle retransmissions of lost packets. In contrast to link layer retransmissions, retransmissions on the transport layer also recover losses occurring within the home network. Retransmission can also be combined with FEC to a hybrid ARQ solution where multiple packets are encoded together to save bandwidth in a multicast retransmission scenario.

Much effort has been spent on developing solutions for loss recovery for multicast video transmission [2]. However, it is not until recently that multicast video transmission is getting widely deployed due to the commercial interest in IPTV. Now we can design the solutions in detail since we know the requirements of the application and the protocols that are being used. In this paper we describe the design and implementation of retransmission functionality for an access node based on currently proposed extensions to RTP, see Section II. The functionality can be enabled individually for each TV-channel in areas where it is needed to solve quality problems for the customers.

In particular, this paper answers the following research questions:

(1) How can the fast RTP retransmission scheme for IPTV be designed and implemented?

(2) What are the parameters that impact the performance of the fast RTP retransmission scheme?

(3) How significant is the impact of the selected parameters on the performance of the fast RTP retransmission scheme?

The next section presents the relevant RTP extensions and the design of the prototype. Section III provides results from our experimental evaluation of the prototype, which shows how to dimension the buffers at the client and the retransmission cache and demonstrate the possible quality improvements. Finally the paper is concluded in Section IV.

Section II answers the research question (1). The research questions (2) and (3) are answered by section III.

II. Design and Implementation

A. RTP Extensions

RTP consists of the real-time transport protocol (RTP) and the Real-Time Control Protocol (RTCP). The receivers send

RTCP receiver reports to the sender or to the multicast group to report some parameters, such as loss rate and jitter. Normally RTCP receiver reports are sent at intervals that depend on the number of users in a multicast group. The purpose is to avoid the scenario that the fraction of the RTCP reports in relation to the total bandwidth is excessively large. However, for IPTV the RTCP receiver reports should not be sent to the whole multicast group for multiple reasons, such as privacy concerns. We consider RTCP feedback that will not be sent to the multicast group, instead it will be sent to the node containing the retransmission cache. This node should filter the reports and possibly aggregate them as described in [3] to provide useful statistics in a scalable way. There is currently an interest in allowing a higher feedback rate, see RFC 4585 [4]. A packet format for RTP retransmission packets is also being standardized by the IETF in RFC 4588 [5]. Those proposals provide the foundation for fast retransmission functionality for RTP. However, for scalability it would not be feasible to use it end-to-end for popular broadcast IPTV channels, therefore we study retransmission functionality located in nodes in the access networks.

RFC4585 defines three different feedback modes. The one we consider is immediate feedback which means that a client can send a feedback message without delay when necessary. Each feedback message will contain one or more generic NACK report(s) which allows multiple lost RTP packets to be reported per feedback message [4].
The retransmissions can either be made in the same session (i.e. on the same IP-address and UDP port number) or on a separate session. The first solution would reduce the number of required ports, but the implementation should not be used in the case of multicast as explained in RFC 4588, therefore we use session multiplexing.

B. Prototype Design

The main components of the prototype are shown in Fig. 1. The retransmission cache stores copies of the RTP packets sent to the clients and stores them in a buffer for a limited time. It also terminates the RTCP sessions from the clients and responds to retransmission requests by retransmitting the required packets if they are still in the buffer. The client is extended compared to a normal RTP client with support for the retransmission requests and buffer handling to insert the retransmitted packets at the right place in the receiver buffer.

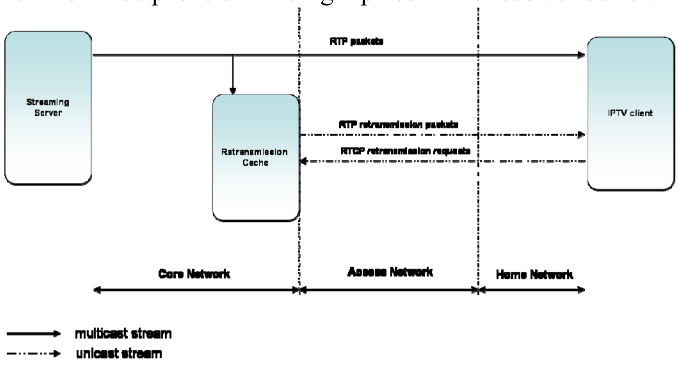

Figure 1. The retransmission cache is implemented in an access node to buffer packets and the client is extended to request retransmission of lost packets.

Before requesting a retransmission the client determines whether it is possible to get the packet in time to be able to use it. This depends firstly on the time it will take to receive the retransmitted packet after a request is sent and secondly on the remaining time before the packet is needed by the application. The first part is solved by estimating the round trip time, including protocol processing delays, by a weighted moving average, similar to the retransmission timeout mechanism used in TCP [6]. The round trip time estimate *srtt* is calculated as follows:

$$srtt_{n+1} = \alpha * RTT + (1-\alpha) * sRTT_n$$

where α in the current implementation is set to 0.6. To reduce the number of unnecessary retransmissions the decision also takes into account the variance in the round trip time, which is estimated as:

$$rttvar_{n+1} = \beta * \left| RTT - sRTT_n \right| + (1-\beta) * rttvar_n$$

where β is set to 0.2. The final decision is based on a retransmission time out (RTO) which is defined as follows:

$$RTO_{n+1} = 2 * RTT_{n+1} + 1.5 * rttvar_{n+1}$$

The remaining time before the packet is required by the application is estimated by the current number of packets ahead of the missing one in the receiver buffer divided by and estimate of the average application read rate.

$$T_{read} = \frac{headOfQueue - missingPacket}{application\_read\_rate}$$

A retransmission request is sent if T_{read} is larger than the *RTO*. The parameters used determines the trade-off between requesting retransmissions that cannot be used because they arrive too late and refraining from requesting packets that could have been used by the decoder. The same *RTO* is also used to determine when to send another retransmission request if the retransmitted packet has not arrived, in case multiple requests for the same packet are allowed.

The retransmitted packets can be sent either by multicast to all clients listening to the retransmission session for the specific channel, or by unicast to the specific client that requested the retransmission. In the multicast case a mechanism that suppresses requests for identical retransmission packets is useful to reduce the amount of feedback. When one client asks for retransmission, the requests for the same packet from other clients are not necessary, because the packet will be retransmitted to those clients anyway. The suppression mechanism would work by letting the IPTV clients wait a random dithering interval to check whether a different client reports the same event, this could be detected by the reception of the retransmitted packet. In particular when losses occur in the core network before reaching the retransmission cache the number of retransmission requests could be a problem, in this case the packet cannot be retransmitted unless there is another retransmission mechanism between the cache and the server.

Both the multicast and the unicast alternatives have their advantages and disadvantages. With unicast the processing load of the retransmission cache will be higher since no messages can be supressed, and the load on common links is higher than for multicast transmission. Multicast transmission has a drawback when there are individual bottleneck links, since these will receive a higher load due to packets retransmitted to other clients. Hence, multicast retransmissions can be beneficial for access networks with a shared medium, e.g. wireless networks or GPON, whereas it is not beneficial for networks where the links are not shared, e.g. DSL. In the prototype the retransmitted packets are sent by unicast.

III. Experimental Evaluation

A. Experiment Setup

The network used in the experiments is depicted in Fig. 2. Network emulator NETEM [7] is used to introduce packet losses and delay. To generate correlated losses trace control for NETEM (TCN) [8] is used. The Distributed Internet Traffic Generator (D-ITG) software package [9] is used in the last experiment to generate competing traffic. The video server sends a MPEG-2 encoded video stream, see e.g. [10], with a frame rate of 25 frames per second, a group of picture size of 12 frames and an average bitrate of approximately 3.6 Mb/s. The delay between the retransmission cache and the clients was set to 10 ms downstream and 2 ms for the upstream direction to reflect a DSL access network (note that the delay for DSL can vary depending on interleaving, but the values are typical examples). Each measurement point in the presented results has been generated by taking averages over 10 test runs and 95-percent confidence intervals are presented in the figures.

After studying very carefully the operation of the fast RTT retransmission scheme, a number of parameters have been identified that can affect the performance of the scheme. These parameters are:

- Size of the retransmission cache and of the client buffer
- Uncorrelated and correlated losses that are experienced by an application
- Congestion induced losses

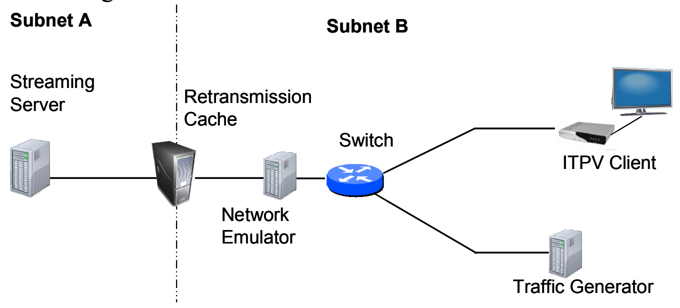

Figure 2. Experiment setup to evaluate prototype.

B. Buffer Dimensioning

1) Retransmission Cache

The size of the buffer in the retransmission cache is an important parameter since a too small buffer would increase the risk that a packet would no longer be available when a request arrives. A too large buffer should be avoided since it would require more memory for the retransmission cache. The goal of this experiment is to evaluate the performance of the proposed scheme when the retransmission cache size and the client buffer sizes are varied. Furthermore this experiment determines the minimum required buffer size at the client and retransmission cache.

Fig. 3 shows the performance of the cache as a function of the size of the buffer. The cache hit ratio is defined as the percentage of retransmission requests that arrive while the requested packet is still available and the retransmission success ratio is defined as the percentage of packets detected as lost by the client that can be recovered. The play-out buffer size of the client is set to a high value (1500 Kbyte) to avoid influencing the retransmission effectiveness. It can be seen that with a buffer size of 25 Kbyte almost all the requested packets can be retransmitted correctly. The required buffer size would increase with the round trip time and the bit rate of the video. The loss rate in the experiment was set to 5%. The reasons that the client does not receive all the retransmitted packets are that some retransmitted packets are lost and others arrive too late.

This experiment has been performed without any competing traffic on the network, therefore it should also be noted that the required buffer size may need to be larger to take into account queuing delays.

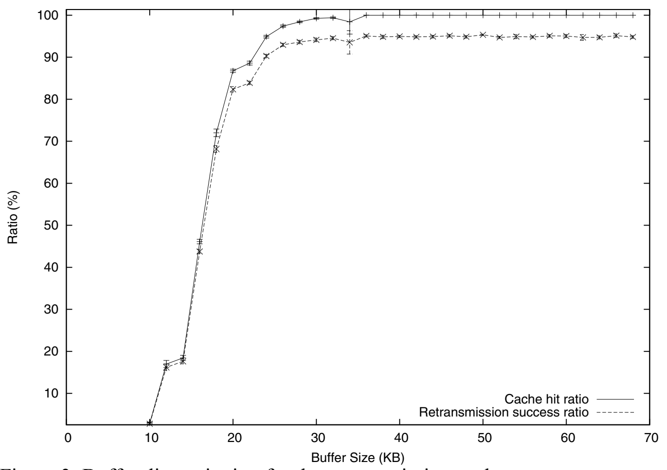

Figure 3. Buffer dimensioning for the retransmission cache,.

Another parameter in the implementation of the retransmission protocol is how many times a client can request retransmission of the same packet. Since a retransmitted packet can also be lost the resulting loss rate is reduced further when the client can request it again if it does not arrive within a given time. The time before requesting another retransmission is based on the estimate of the mean and variance of the retransmission time.

Fig. 4 shows the cache hit ratio and the retransmission success ratio as a function of the buffer size when three retransmission requests are allowed. It can be seen that the success ratio continues to increase when the buffer size is increased above 25Kbyte in this case, since the retransmission requests can arrive with larger delay. The remaining uncorrected loss rate is also lower than with a single retransmission request per packet. The figure also shows that the retransmission success ratio is higher than the cache hit ratio, this is because multiple requests may be sent for a packet that are not in the buffer, hence reducing the cache hit ratio.

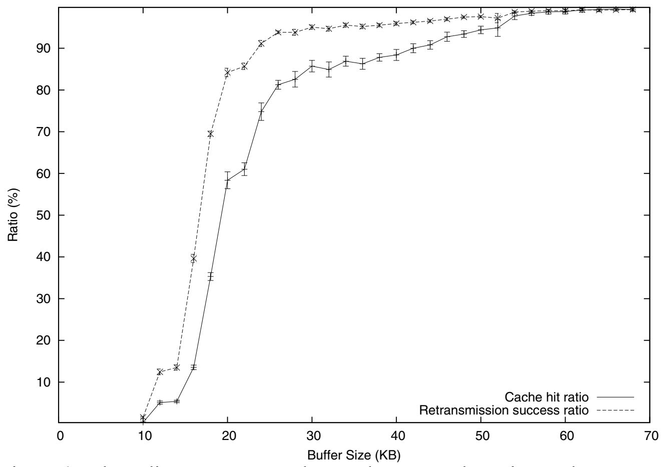

Figure 4. When clients can request lost packets up to three times a larger buffer improves performance.

2) *Client Buffer*

The size of the client buffer is also an important parameter. Fig. 5 shows how the retransmission works for different sizes of the client buffer. It can be seen that the retransmissions are successful and the loss rate strongly reduced, but when the buffer is too small the frames are delayed which results in non fluent video playback. Hence, the buffer would need to be in the order of 500 Kbyte for the given video bit rate and round trip time. As opposed to the buffer in the retransmission cache the client buffer is not defined as a maximum amount of memory available. Instead the client buffer size is the buffer fill level that the retransmission logic builds up before the application starts decoding.

Also without the retransmission functionality the receiver needs to build up a buffer. The size of the playout buffer in general depends both on the jitter induced in the network, the traffic shaping in the sender and how the application reads out the data from the buffer. In the prototype the video server sends a video stream shaped to a constant bit rate. The application reads out the packets from the buffer when it needs them in the decoding process, this is not a constant rate since the frames of the video coding varies in size, e.g. the I-frames are larger than P- and B-frames. Therefore, the client requires a buffer size of a few hundred kilobytes also without the retransmission functionality.

The additional buffer size required when retransmission is used can be considered to be upper bounded by the buffer size in the retransmission cache, since that corresponds to the additional delay that can be introduced for retransmitted packets. However, the required additional buffer can also be smaller since the playout buffer can be used also for retransmission purposes. This can be seen from the gentle degradation as the buffer size is reduced.

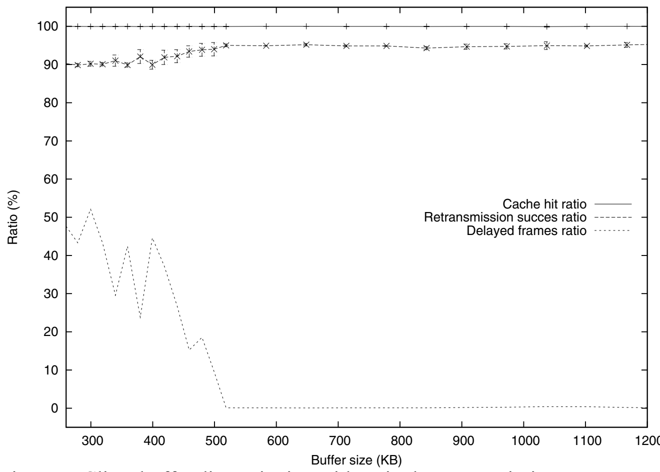

Figure 5. Client buffer dimensioning with a single retransmission attempt allowed.

C. *Loss Recovery*

1) *Uncorrelated Losses*

The goal of this experiment is to evaluate the performance of the proposed scheme under different uncorrelated and correlated loss scenarios in an uncongested access network. The gain from the retransmission functionality can be measured by the reduction in loss ratio that the application experiences. In Fig. 6 the remaining loss ratio is plotted as a function of the loss probability introduced by the network emulator. In this experiment the loss process is uncorrelated. Already with a single retransmission attempt the remaining loss ratio can be reduced for example from three percent to 0.1 percent. With increasing network loss the remaining loss ratios increase and the results show that with two retransmission attempts the loss rates can be significantly reduced, but allowing a third retransmission does not reduce the loss rate notably. This depends to a large extent on the size of the client buffer, when too many retransmissions are allowed the retransmitted packets may no longer be useful to the application. In this experiment the client buffer size was set to 780 Kbyte, and the RTO as defined in Section II; with these settings the third retransmission does not provide much additional benefit.

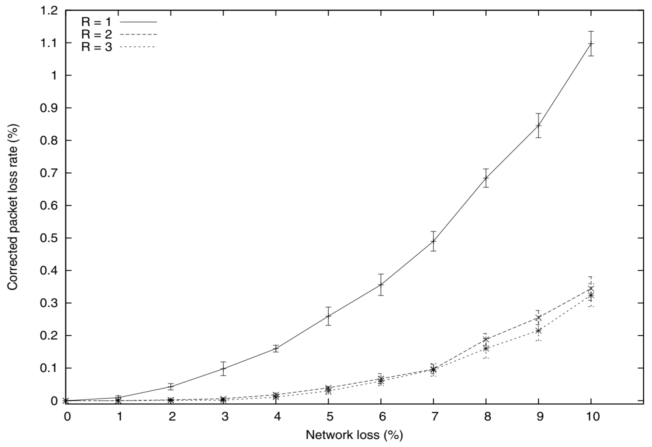

Figure 6. Recovery efficiency for different number of retransmission attempts.

2) *Correlated Losses*

In many network scenarios the packet loss process is correlated. This can have implications on the efficiency of different loss recovery techniques. To evaluate the efficiency of the retransmission scheme in case of correlated losses we use an experiment where the losses are generated by a Gilbert-Elliot model, i.e. a two state Markov chain where packets are lost in one of the states and not lost in the other one. When a packet is lost the probability of remaining in the loss state and losing the next packet is 0.8, the probability of being in the loss state and the good state depends on the average loss rate.

In this simulation the parameters for the retransmission feasibility check is reduced to 1.4*RTT instead of 2*RTT which makes also a third retransmission feasible. The resulting loss reduction for one, two and three retransmissions is shown in Fig. 7. The results show that the remaining loss rate is higher than they were with independent losses; hence the retransmissions are a bit less effective. One reason for this is that losses are detected later due to the large gaps, therefore there is less time left for retransmissions. It can also be seen that the third retransmission provides a further loss reduction in this case.

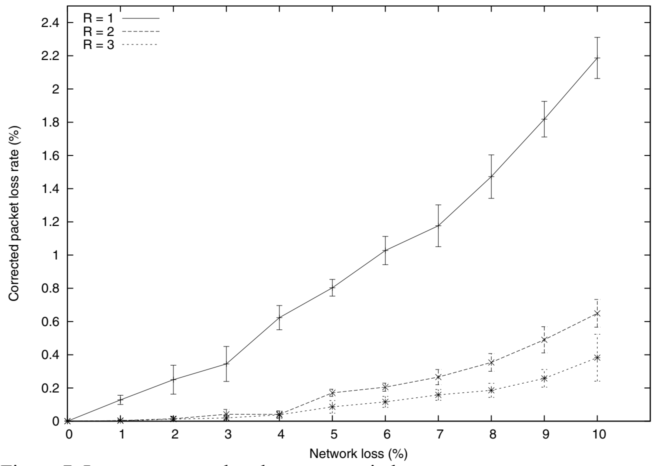

Figure 7. Loss recovery when losses occur in bursts.

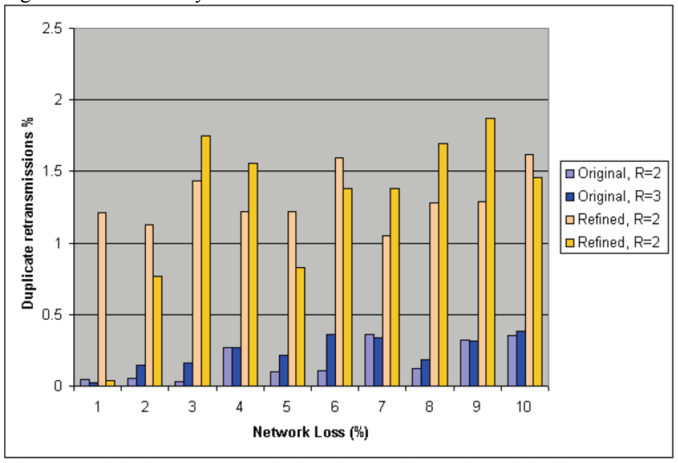

Figure 8. With shorter retransmission timeout the number of duplicate retransmission packets increases.

The drawback of a the new settings for the retransmission feasibility check is that some packets will be retransmitted multiple times, even though the first retransmitted packet arrived correctly. This can be seen in Fig. 8, where the refined results refer to the shorter timeout value (1.4*RTT) and the original is 2*RTT. The number of unnecessary retransmissions received by the client increases from under 0.5 percent to in the order of one percent due to the faster retransmission for new retransmissions, which illustrates the tradeoff between lower experienced loss rate and wasted bandwidth.

Another concern when using retransmission is the increased traffic load caused by the retransmission packets and the retransmission requests. Since the video stream is unidirectional the downstream is more likely to be a bottleneck than the upstream, therefore we only consider the extra traffic in the downstream. Fig. 9 shows the increase in the bandwidth when retransmission is used compared to the video stream alone, as a function of the network loss rate. The bandwidth increase is approximately proportional to the loss rate, as expected.

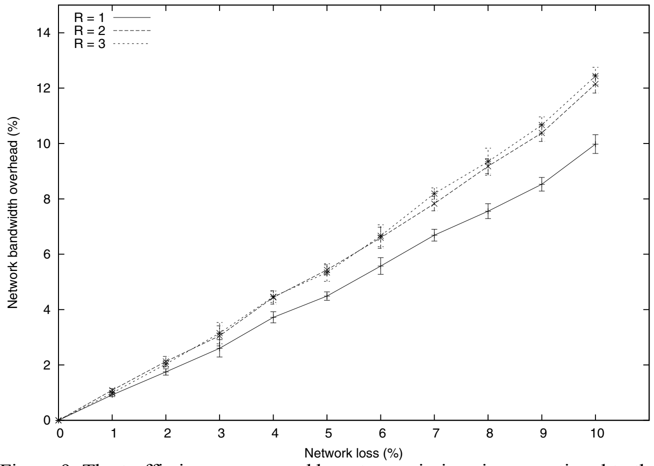

Figure 9. The traffic increase caused by retransmissions is proportional to the loss rate.

D. *Congestion Induced Losses*

The goal of this experiment is to evaluate the performance of the proposed scheme when losses in the network are occurring due to congestion. When losses occur due to congestion the efficiency of retransmission is less obvious than when losses are independent of the traffic load. Therefore, we investigate a scenario where the video traffic has to share the available network capacity with a TCP session. The congestion control of TCP increases the rate until losses occur to probe how much available capacity there is, hence it will cause losses both to itself and to the video traffic. When packets are retransmitted the load increases and the loss rate increases further. In Fig. 10 it can be seen how an FTP session is started while the video transmission is ongoing. The congestion control of TCP fills the remaining capacity of the link which causes some packet loss, it can be seen that retransmissions are also being sent when the FTP session is active.

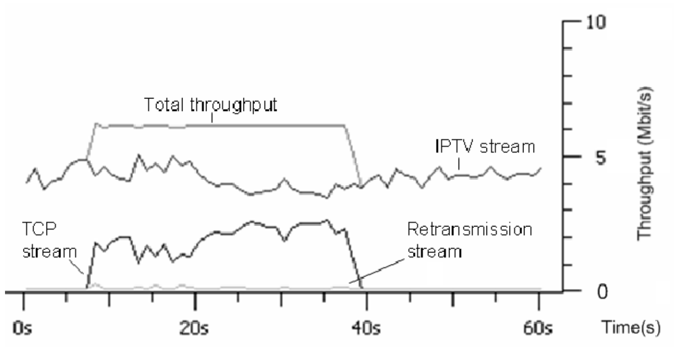

Figure 10. A TCP session causes congestion during 30 seconds, which causes some losses and retransmissions.

Fig. 11 shows the resulting loss rates when retransmissions are not enabled and when they are enabled. As expected the retransmission contributes to increase the network loss rate compared to when there are no retransmissions. However, the results also show that the application loss rate is lower when retransmissions are enabled. Hence, there is a gain of using retransmission also when losses occur due to congestion.

It can be argued that the other applications, in this case the FTP session, will suffer from the bandwidth taken by retransmissions. There may be concerns about stability and TCP friendliness since the sending rate increases when losses occur. However, the IPTV stream is not congestion controlled in the first place, so it makes little sense to consider TCP-friendliness. For the overall utility of the network it is better that the IPTV session is delivered with good quality at a slightly increased sending rate than that it is delivered with bad quality at its nominal sending rate.

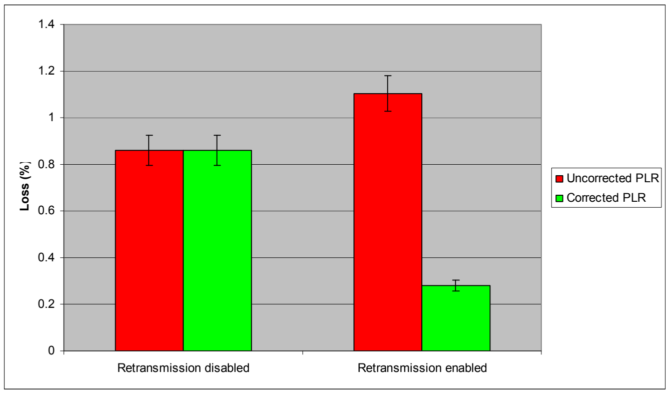

Figure 11. With retransmissions enabled the raw loss rate increases due to the extra load, but the application experiences lower loss.

IV. Conclusion

Referring to the three questions in the introduction, we have first described how a fast RTP retransmission scheme can be implemented using extensions to RTP. Secondly, we have identified the parameters that influence the performance, for example when to request a retransmission, how many retransmission requests should be allowed and how to choose the buffer sizes in the retransmission cache and the client. Thirdly we have used experiments to quantify how important those parameters are and described the trade-offs involved. The results show that the loss rate can be reduced substantially, which would result in much higher video quality for the viewers.

Deployment of this type of solution requires some added complexity in the clients and the nodes containing the retransmission caches. The buffer requirements in the retransmission cache are moderate, and if the retransmission scheme is enabled only when it is actually needed by the customers the increased processing load can be acceptable. How many users that is reasonable to handle per retransmission cache is left for future work.

References

[1] M. Luby et al, “The use of forward error correction in reliable multicast”, RFC 3453, Dec 2002.

[2] G. Carle, E.W. Biersack, “Survey of error recovery techniques for IP-based audio-visual multicast”, IEEE Network, vol 11, issue 6, Nov. 1997, pp. 24-36.

[3] J. Ott, J. Chesterfield, E. Schooler, “RTCP Extensions for Single-Source Multicast Sessions with Unicast Feedback”, Internet draft (work in progress), March 2007, URL: http://tools.ietf.org/html/draft-ietf-avt-rtcpssm-13.

[4] J. Ott et al. “Extended RTP profile for real-time transport control protocol (RTCP)-based feedback (RTP/AVPF)”, RFC 4585, July 2006.

[5] J. Rey et al. “RTP retransmission payload format”, RFC 4588, July 2006.

[6] V. Paxson, M. Allman, “Computing TCP’s Retransmission Timer”, RFC 2988, Nov. 2000.

[7] Network Emulation with NetEm, 4 2005. URL: http://www.linux-foundation.org/en/Net:Netem

[8] A. Keller, “TCN trace control for netem” 2008. URL: http://tcn.hypert.net/

[9] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre, “D-itg distributed internet traffic generator” in QEST. IEEE Computer Society, 2004, pp. 316-317. URL: http://csdl2.computer.org/comp/proceedings/qest/2004/2185/00/21850316.pdf

[10] ITU-T Rec. H.222.0 | ISO/IEC 13818-1:2000, “Generic coding of moving pictures and associated audio information, Part 1: Systems”, May 1999.