Scalable Packet Loss Recovery for Mobile P2P Streaming

Jani Peltotalo[1], Jarmo Harju[1], Lassi Väätämöinen[1],
Imed Bouazizi[2], Igor D.D. Curcio[2], and Joep van Gassel[2]

[1] Tampere University of Technology, Department of Communications Engineering
P.O.Box 553, FI-33101 Tampere, Finland
{jani.peltotalo,jarmo.harju,lassi.vaatamoinen}@tut.fi
[2] Nokia Research Center
P.O.Box 1000, FI-33721 Tampere, Finland
{imed.bouazizi,igor.curcio,joep.van.gassel}@nokia.com

Abstract. In a real-time peer-to-peer streaming system peers in the overlay network may arrive and depart in a very dynamic fashion especially in mobile environment. This manifests itself by a sudden uncontrolled disappearance of a sender and, as a consequence, packets being lost at the receiving side. To ensure seamless media playback the data should not have interruptions. Therefore, if some data packets are missing those should be requested and retrieved from other peers before the playback point reaches the gap in the reception buffer. This paper presents a scalable two-stage packet loss recovery mechanism for a real-time peer-to-peer streaming system using RTCP and RTSP.

Keywords: Real-Time, Peer-to-Peer, Streaming, Packet Loss Recovery, RTP, RTCP, RTSP.

1 Introduction

In traditional streaming applications based on the Real-time Transport Protocol (RTP) / RTP Control Protocol (RTCP) [7], one or more receivers are connected to a single sender. Although there can be multiple senders in a multi-party communication, a receiver of a particular RTP session is only retrieving data from a single sender. In such applications, failures in the network path between sender and receiver can cause packet losses. Special mechanisms are available for the retransmission of those lost packets. This procedure consists of signalling the losses by the receiver and retransmitting the lost packets by the sender. One method to signal packet losses to the sender is using the Generic NACK RTCP message, as specified in [3]. In addition [6] specifies how the lost packets can be retransmitted by delivering them in a separate stream.

In [4] results from experimental test with some of the currently existing P2P streaming application, such as Octoshape [2] and SopCast [10], shows that improvements are needed to enhance the mobile usage. Our real-time Peer-to-Peer (P2P) streaming system [5] contains several important improvements, like small

E. Osipov et al. (Eds.): WWIC 2010, LNCS 6074, pp. 107–120, 2010.

ten seconds initial buffering time, ten seconds buffer size due to the RTP usage, and the partial RTP stream concept where the original RTP sessions related to a media delivery are split into a number of so-called partial RTP streams which allow a single media stream, like an audio or video component, to be received simultaneously from multiple senders.

Because these peers may join and leave the service at their own will, streams being sent to a receiver may be temporarily interrupted, introducing another reason for packet losses in a P2P streaming system. Hence, in a multi-sender P2P streaming environment two different causes, due to which packet loss may be experienced, can be distinguished: (a) network failure, and (b) peer churn. Upon detection of packet loss by a receiving peer, the peer does not necessarily know which one of the two options described above is causing the experienced packet loss. Therefore, the traditional RTCP-based packet loss signalling is insufficient in the P2P streaming case. This is due to the fact that when a sender completely leaves the network, it does not make sense to keep sending Generic NACK RTCP messages to signal lost packets since they will never be received by the departed sender.

An additional requirement is that the packet loss recovery mechanism should be scalable. Once a single connection becomes unavailable, all the peers in an isolated region may start signalling packet losses or explicitly start to request retransmissions. This occurs when one peer along the path back to the original data source departs from the overlay network. In that case, not only peers directly receiving from the departed peer, but also all other peers down the supply chain would start signalling packet losses or requesting retransmission of the same data. Such message propagation throughout the system is undesirable and should be avoided, or kept to the minimum, in order to allow the packet loss recovery mechanism to scale to large network sizes.

The structure of the remainder of this paper is as follows. Packet loss recovery mechanisms based on RTCP and Real Time Streaming Protocol (RTSP) [8] are presented in Sections 2 and 3, respectively. Then, a scalable two-stage packet loss recovery mechanism combining RTCP and RTSP is presented in Sections 4 and 5. After that, results from the performance experiments are presented in Section 6. Finally, Section 7 gives a concluding summary and a look into future developments.

2 RTCP-Based Packet Loss Recovery Mechanism

Generic NACK RTCP messages are used to signal packet losses from the receiver to the sender in a traditional streaming application. A number of variations on how to use RTCP in the P2P streaming system for this purpose is illustrated in Fig. 1, where `Peer Y` is receiving *audio*, *video partial 0* and *video partial 1* from the `Audio`, `Video 0`, and `Video 1` peers, respectively. In this particular example, `Peer Y` has detected packet losses from the `Video 0` sender. Using RTCP the peer may signal packet losses to other peers using RTCP in the following three ways:

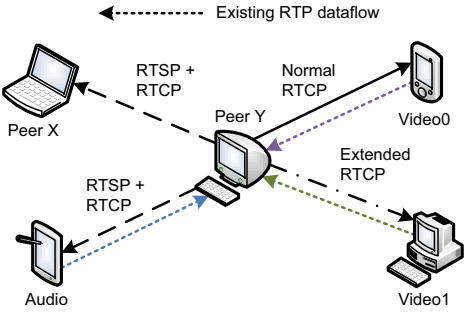

Fig. 1. Packet loss recovery mechanism based on RTCP

1. *Normal RTCP*, as specified in [3] by sending Generic NACK RTCP Receiver Reports (RRs). The receiver, i.e., `Peer Y` in Fig. 1, uses RTCP RRs to signal the losses to the sending peer from which it was expecting to retrieve the lost packets, i.e., `Video 0` sender in Fig. 1. This can be considered as normal operation according to [3].
2. *Extended RTCP*, in which case the receiver uses RTCP RRs to signal the losses to a peer that was not serving the missing packets in the first place. However, the peer receiving the RTCP messages was serving a different partial from the same RTP session, i.e., `Video 1` sender in Fig. 1. Note, that this extends the scope of the original RTCP specification beyond its normal use.
3. *RTSP + RTCP*. Since RTCP can only be used in the presence of an RTP connection, RTSP must be used to set up an RTP session between peers if no media stream was set up beforehand, i.e., `Audio` sender and `Peer X` in Fig. 1. In this case the sole purpose of providing an RTP stream is the retransmission of lost packets.

To summarize, RTCP is only suitable for requesting retransmission of data in the case where the sending peer is still available. Otherwise, a new stream would have to be set up using RTSP. Furthermore, it should be noted that RTCP is already in place in most implementations so this is a fairly low-impact way of signalling packet losses which is completely in line with current specifications. The amount of overhead is small, since the Generic NACK messages are sent along with the normal RTCP RRs in compound RTCP messages. However, a peer does not explicitly request retransmission; it only signals which packets have been lost to the sender, and it is up to the sender to decide how to deal with this information.

3 RTSP-Based Packet Loss Recovery Mechanism

Alternatively to the RTCP-based packet loss signalling, the `RTSP PLAY` message can be used for requesting retransmission of lost packets. However, the current

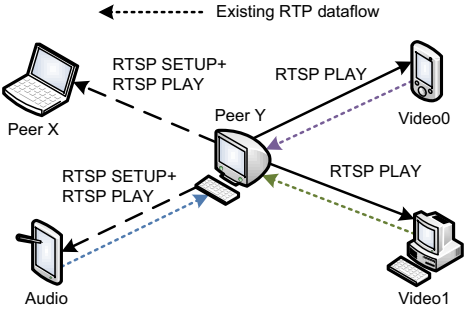

Fig. 2. Packet loss recovery mechanism based on RTSP

syntax of the `Range` header field does not allow requesting retransmission of individual packets or a limited set of packet ranges; it allows time-based ranges but these do not suffice to uniquely identify single packets. This paper proposes the necessary header field extension to make the RTSP-based packet loss signalling possible. The format of a `Packet-Range` header field in ABNF is given below. This format overrides the format specified in [5], where the `Packet-Range` header field was used to signal the play-after value to the replacement peer, without the possibility to specify any missing packet preceding the last received packet from the departed peer.

```
Packet-Range = "Packet-Range:" SP 1*range-specifier CRLF
range-specifier = 1*DIGIT ["-" [1*DIGIT]] ";"
```

In Fig. 2 `Peer Y` has experienced packet losses from the `Video 0` sender and may request retransmission of the lost packets from alternative source peers in the following two ways by means of RTSP:

1. *RTSP PLAY.* This is the case where a suitable media stream was already set up using RTSP beforehand, i.e., `Video 0` and `Video 1` senders in Fig. 2. The peer sends an `RTSP PLAY` message with a `Packet-Range` header field containing sequence numbers of the missing packets directly to the candidate source peer.
2. *RTSP SETUP + RTSP PLAY.* This is the case where a new peer, which was not serving a suitable media stream that could be used for retransmission, i.e., `Audio` sender or `Peer X` in Fig. 2, is selected among the candidate source peers. In this case an `RTSP SETUP` message is first used to create a connection between peers, and an `RTSP PLAY` message with a `Packet-Range` header field is subsequently used to request the lost packets.

The candidate source peer will respond with a `200 OK` message if all of the requested data is available and the requested packets will be streamed to the requesting peer. A `206 Partial Content` message will be sent if the requested data is partially available, and the available packets will be sent to the requesting peer. The `206 Partial Content` message contains also information on what

parts are actually streamed using the `Packet-Range` header field. Otherwise, a `204 No Content` message will be returned if the candidate source peer does not have the requested packets, or a `453 Not Enough Bandwidth` message if the candidate source peer does not have enough bandwidth available to successfully handle the given request.

Note that in order to speed up the reconnecting process, the `RTSP SETUP` and `RTSP PLAY` messages could be combined using pipelining as specified in [9] and [1]. In relation to the earlier described RTCP-based method, the RTSP-based method generates more message overhead and latencies. However, because of the request and response nature of RTSP signalling, a peer will always receive information about the sender's capability to actually retransmit the requested packets.

4 Scalable Two-Stage Packet Loss Recovery Mechanism

To ensure seamless media playback the data should not have interruptions. Therefore, if some data packets are missing, those should be requested from other peers before the playback point reaches the gap in the reception buffer. In order to allow for scalable and efficient retransmission of lost packets in the presence of network failure and unexpected peer churn, a simple two-stage packet loss recovery mechanism is proposed in this section. Scalability is necessary to prevent a ripple through effect of retransmission requests and redundant retransmissions in case many peers start simultaneously requesting lost packets due to a single cause, like a failing or departed peer. The proposed packet loss recovery mechanism consists of two stages:

1. Use RTCP in the normal and extended way to signal packet losses to the sending peers.
2. In case lost packets are not received during a certain time-out value T_W, use RTSP to set up new connections and to request the retransmission of the missing packets.

In combination with the above mentioned two stages, a special signalling of pending retransmission requests and resolved packet losses can be applied to enhance the scalability of the proposed packet loss recovery mechanism.

4.1 Stage One

Every RTP session, such as audio, video or subtitle streams of the entire multimedia session is split into smaller pieces, each consisting of a group of RTP packets, along the time axis. Every piece has a fixed duration T_P which is expressed in time. The N partial streams are constructed by assigning pieces sequentially one by one into the partial streams *0...N-1* and then continuing again with partial stream *0*, etc.

The time-out value T_L, defined by (1), is signalling the amount of time that is waited until the packet is considered lost. This value consists of the normal network delay T_N, that can be calculated for example from RTSP request

response pairs, and an extra time T_E given to peers to patch their missing packets to avoid overloading the network with retransmission requests for packets that might still be forwarded in the network. If the missing packet is the last packet of the piece, then the time $T_B = T_P * (N - 1)$ between two pieces belonging to the same partial RTP stream will be also added to the time-out value.

$$T_L = \begin{cases} T_N + T_E, & \text{not last packet in the piece,} \\ T_N + T_E + T_B, & \text{last packet in the piece.} \end{cases} \quad (1)$$

After the time-out value T_L has expired, the signalling of packet losses using RTCP is started. Firstly, the packet losses are signalled back to the peer it was expected to receive the missing packets from. Alternatively or simultaneously, a similar RTCP RR may be sent to other peers serving different partial RTP streams from the same RTP session.

4.2 Stage Two

If the packets that have been signalled as lost using RTCP do not arrive within a certain waiting time T_W, the receiving peer selects a new candidate source peer and starts to request retransmission of the lost packets by means of RTSP. This occurs for instance when a sending peer departs without signing off from the network, for example due to malfunctioning, or when the used network path is broken. Note that in case a source peer departs in a controlled way, the peer will also reconnect to an alternative source peer and resume playback from the point of interruption as specified in [5]. However, this is considered to be normal operation of the peer-to-peer streaming system. If the waiting time T_W is set to zero, then the peer will directly go to the stage two, of course after the time-out value T_L is expired, and start to request missing packets from the candidate source peers using RTSP.

The candidate source peers can be peers that are already serving the receiving peer with other media streams of the same service, for example the `Audio` peer in Fig. 2, or completely different peers, like `Peer X` in Fig. 2. Note however, that these cannot be peers that are already serving other partials from the same RTP session, since in that case RTCP RRs should be used as described in stage one. Retransmissions from the new peers are explicitly requested by setting up a new RTP connection using `RTSP SETUP` and `RTSP PLAY` messages.

The new RTP connection can either be set up permanently, or used only to retrieve missing packets. In the former case the old existing RTP connection is discarded and playback resumes with the new peer possibly after individual missed packets have been received. In the latter case, the new connection will be torn down using the `RTSP TEARDOWN` message after the lost packets have been retransmitted. In order to improve the speed of retransmissions, backup RTSP connections can be kept open, without actual streaming taking place, just for the purpose of requesting retransmission of lost packets in case other peers fail. This allows faster error recovery, since the set up time is eliminated from the retransmission procedure. Alternatively, similar performance improvements can also be realized by pipelining the `RTSP SETUP` and `RTSP PLAY` messages.

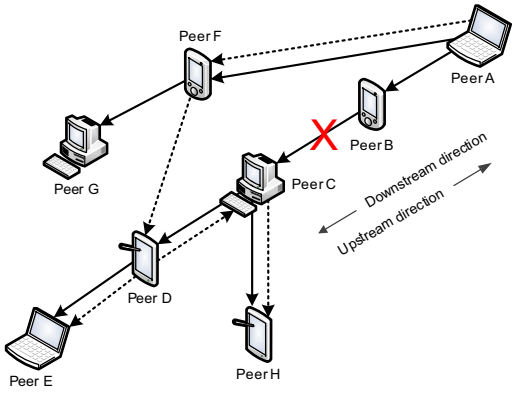

Fig. 3. Recovery of broken streaming path of a partial RTP stream

4.3 Avoiding Retransmission Request Overload

In order to avoid multiple peers, downstream from the point of failure in a broken path, from signalling lost packets due to the same root cause, such as a malfunctioning peer, or a peer departed in an uncontrolled way or cut off from the network connection, a special signalling mechanism is devised. The proposed mechanism includes two aspects. The first aspect is the signalling of pending retransmission requests downstream, i.e., in the direction of the data flow from the original data source to the leaf nodes. Hence, in the absence of packets, the information is signalled to indicate to the receiving peers that the sender is aware of the losses. Secondly, it includes the signalling of the recovered packets both downstream and upstream. This information can be utilized by the recipients to efficiently reconnect to other peers which have signalled that they have resolved the problem.

Fig. 3 illustrates a delivery flow of a particular partial RTP stream using solid edges. In the figure, `Peer B` is malfunctioning and causing `Peer C`, `Peer D`, `Peer E` and `Peer H` further downstream to experience packet losses. Note that the packet losses could also be caused by a failing network link between `Peer B` and `Peer C`. This situation may cause all of these peers to start signalling packet losses or requesting retransmissions and, as a consequence, this will flood the network with redundant retransmissions of the same lost packets. If packet retransmission is handled in such uncoordinated way, the efficiency and scalability of the system is seriously jeopardized. Therefore, the peers will signal pending retransmission requests downstream and recovery of lost packets both up- and downstream. This is achieved by enhancing the RTCP feedback messages specified in [3].

The common packet format for feedback messages is depicted in Fig. 4. Both pending retransmission request and recovered packet loss messages are using RTPFB value (205, transport layer feedback message, specified in [3]) in the Packet Type (PT) field. The first synchronization source (SSRC) identifier

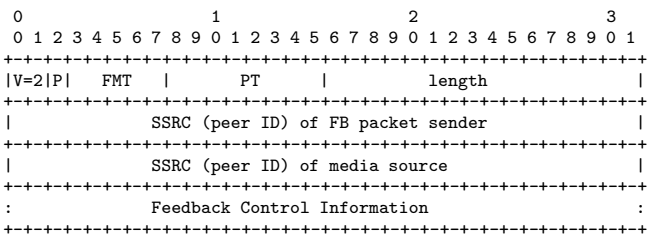

Fig. 4. Common packet format for feedback messages

identifies the sender of the feedback message and the second SSRC identifies the original sender of the media data.

A single feedback control information block can be repeated multiple times and has the format shown in Fig. 5. The Packet IDentifier (PID) specifies the sequence number of the first lost or recovered packet. The feedback message type (FMT) will be used to distinguish between pending retransmission request and recovered packet loss messages. The semantics of the feedback control information can be given either in the NPLR field (Number of Packets Lost or Recovered) or in the BLP field (Bitmask of following Lost Packets). The NPLR field represents the number of packets lost or recovered from PID onwards excluding the packet with PID itself. The BLP field represents a bit mask identical to the syntax and semantics of the BLP field used in the Generic NACK message specified in [3].

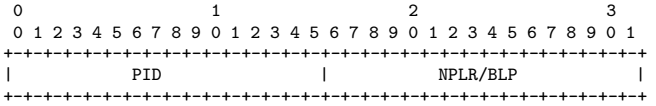

Fig. 5. Feedback control information block

It is the responsibility of a receiving peer to aggregate this information from the incoming streams and send it out to the outgoing streams. This is especially important if the configuration of the partial streams is not constant within the service. In practice, this is the case when a particular media stream is portioned into a different number of partials on the incoming and outgoing connections of a peer. The explicit signalling of packet loss to downstream peers also reduces the time it takes for a particular peer to detect a packet loss. This is due to the fact that in the absence of packets, signalling is available to indicate these losses. Hence, the peer does not need to wait too long before it can conclude that packets have been lost somewhere upstream.

In case a peer discovers pending retransmission requests, it may either wait until these lost packets arrive from the sending peer, or it may choose to start its own packet loss recovery mechanism. The former occurs when the upstream peer

has found an alternative source or the original sending peer has recovered from the problematic situation. Once any peer has successfully recovered lost packets by reconnecting to a new peer, it will signal this both up- and downstream. This improves the self-healing capability of the peer-to-peer streaming system. In this way, the overlay may be reorganized jointly by the distributed peers in a coordinated effort. This is illustrated in Fig. 3, where `Peer D` discovers an alternative source (`Peer F`) before the sending peer (`Peer C`) solves the problem. After `Peer D` has signalled packet loss recovery upstream to the `Peer C`, `Peer C` may for instance decide to switch roles and use `Peer D` as a new source for this particular partial RTP stream. This means in practice that a new data flow, indicated by the dashed edges in the figure, is constructed jointly by the involved peers.

5 Peer Operation

The operation of a peer using the two-stage packet loss recovery mechanism including signalling of pending retransmissions and recovered packets is depicted in detail in Fig. 6. A peer receives RTP packets in a normal manner (block 1). When it determines that packets have been lost (block 2), i.e., the time-out value T_L for some missing packets has expired, it enters to the first stage of the two-stage packet loss recovery mechanism.

In the first stage, an additional scalability mechanism utilising active and passive modes can be used to avoid overload of packet loss signalling. In the passive mode a peer may wait for the stream path issues to be resolved, while in the active mode (block 3) it tries to actively resolve the stream path issues. If the peer is in the passive mode, it waits for a period of time T_D (block 4), defined by (2), to allow the stream path issues to be resolved. $Peer_{position}$ is the position of the peer in the path from the original data source. This position can be indicated for example by the Contributing Source (CSRC) header field in the RTP packet. Max_{depth} is the maximum allowed depth for the stream path set by the P2P streaming system operator, and in the current implementation it is set to 15. In (2), $\alpha \in (0, 1)$ is a random coefficient to diversify the delay time among peers in the same level of the stream path. D_{max} is the maximum delay which still allows to recover from the packet loss before the playback point reaches the gap in the reception buffer and is dependent on the used initial buffering time.

$$T_D = \frac{Peer_{position}}{Max_{depth}} * (1 - \alpha) * D_{max} \tag{2}$$

After T_D has elapsed, the peer in the passive mode checks whether an indication of a pending retransmission request has been made (block 5). If such an indication has been received for the missing packets in question, the peer waits for another small period of time (block 6) to allow an upstream peer to retrieve and forward the lost packets. If the lost packets are received (block 7), the peer can stop the packet loss recovery mechanism for packets in question. If the lost packets are not received, the peer returns to the block 6 and waits again for the small period of time. This is continued until the playback point reaches the

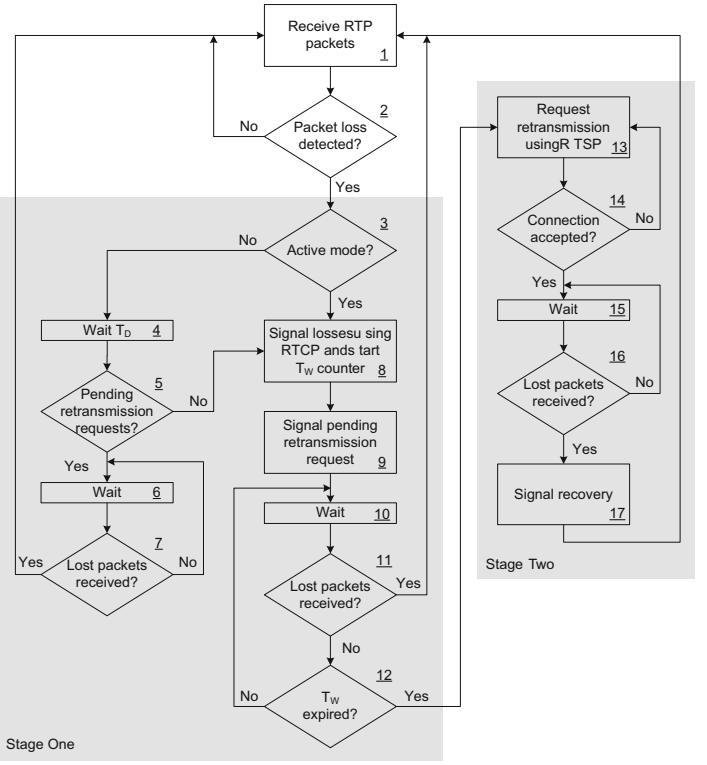

Fig. 6. Two-stage packet loss recovery mechanism

missing packets. On the other hand, if pending retransmission requests have not been signalled (block 5), the peer proceeds to signal an indication of lost packets using RTCP (block 8) to the upstream peer. Similarly, if the peer is in an active mode (block 3), the peer proceeds to the block 8 and starts the T_W counter.

Next, the peer signals to the downstream peers an indication of a pending retransmission request (block 9). The peer then waits for a small period of time (block 10) and checks whether the lost packets have been received (block 11). If the packets have been received, the peer can again stop the packet loss recovery for the packets in question. On the other hand, if the lost packets have not been received, the peer checks whether a threshold period of time T_W has expired (block 11). If the threshold period of time has not expired, the peer repeats blocks 10, 11 and 12. If the threshold period of time has expired, the peer enters to the second stage of the two-stage process.

In the second stage, the peer sends a request for RTP packet retransmission using RTSP (block 13). Next, the peer will discover whether an RTSP connection has been accepted (block 14). If the connection is not accepted, the peer returns to block 13 until the connection is accepted. Once the connection is accepted, the peer waits (block 15) and determines whether the lost packets have been received (block 16). Once the lost packets are received, the peer signals an indication of

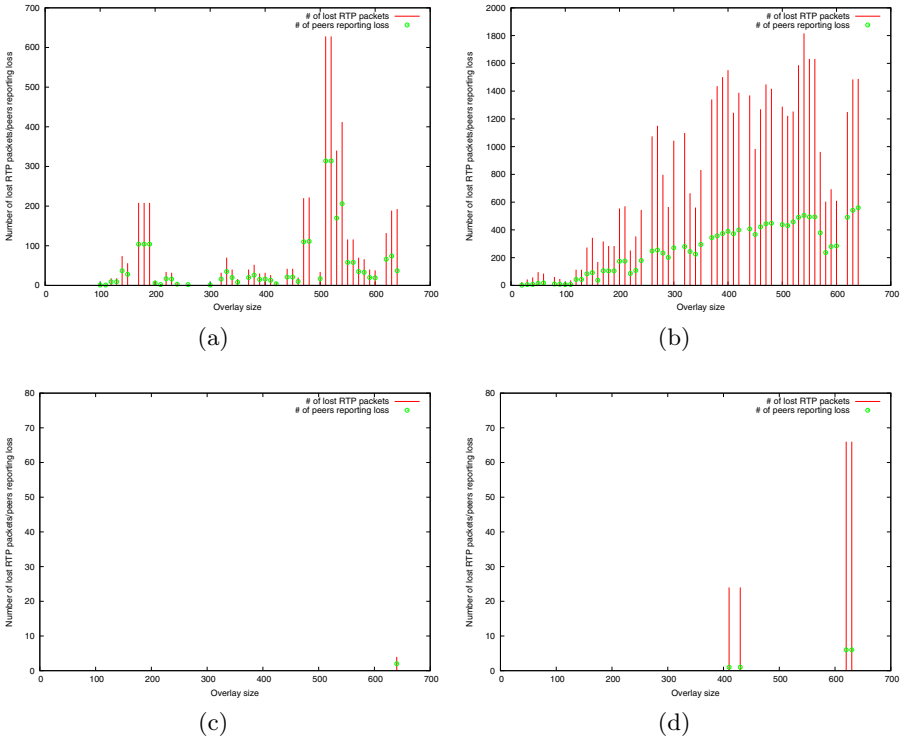

Fig. 7. The number of lost packets with churning peers, (a) audio stream without loss recovery, (b) video stream without loss recovery, (c) audio stream with loss recovery, and (d) video stream with loss recovery

the packet loss recovery both to up- and downstream peers (block 16), and the peer can stop the packet loss recovery for the packets in question.

6 Performance Evaluation

Currently our P2P streaming system contains only an RTSP-based packet loss recovery mechanism, specified in Section 3, in addition to the peer replacement presented in [5]. So the results presented in this section illustrate the benefits of that kind of packet loss recovery, but the overall system performance could be enhanced with a complete two-stage packet loss recovery mechanism.

As in [5], a laboratory network environment with 17 desktop PCs has been utilized to evaluate the operation of the system with and without the RTSP-based packet loss recovery mechanism. 16 hosts were used to run 40 peers in each host together with one host acting as an original data source. The length for one live streaming service was roughly one hour, and T_P was set to 100 milliseconds and N was set one for audio stream and, correspondingly, 400 milliseconds and

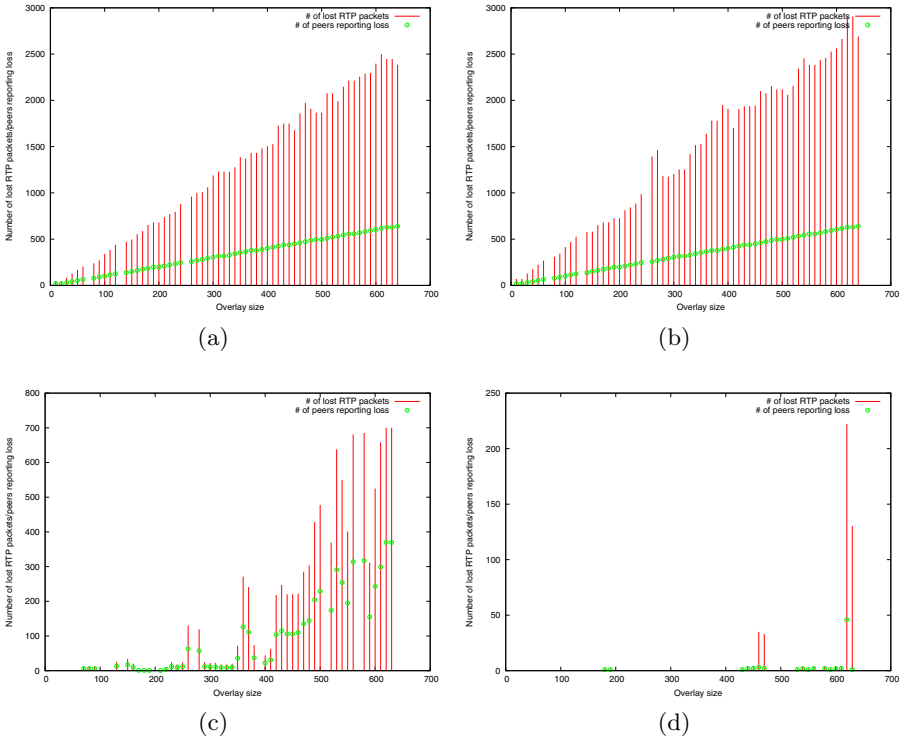

Fig. 8. The number of lost packets with a 1% uniformly distributed packet loss, (a) audio stream without loss recovery, (b) video stream without loss recovery, (c) audio stream with loss recovery, and (d) video stream with loss recovery

four for video stream. The maximum cluster size was set to 70 peers, and peers were started in 40 cycles with five seconds starting interval: first one peer was started at each host, then a second one and so on. To simulate churning caused by the real mobile nodes, a timer functionality which randomly shuts down and restarts nodes was used. After a peer had joined to the service, it stayed randomly up to 30 seconds in the service and then left the service and joined back randomly after one to ten seconds.

The number of lost packets per peer as a function of the overlay size with and without RTSP-based packet loss recovery are shown in Fig. 7 (with churning peers) and Fig. 8 (with a 1% uniformly distributed packet loss). From the figures we can see that the improvement with a packet loss recovery is remarkable. With churning peers, only few packets remain lost with the packet loss recovery. This loss is due to the fact that also some of the selected new source peers departs before sending all of the requested missing packets and the small ten seconds initial buffering time does not allow requesting some of the missing packets again. With a 1% uniformly distributed packet loss, retransmitted packets are also affected by the packet loss and causes some amount of packets to remain

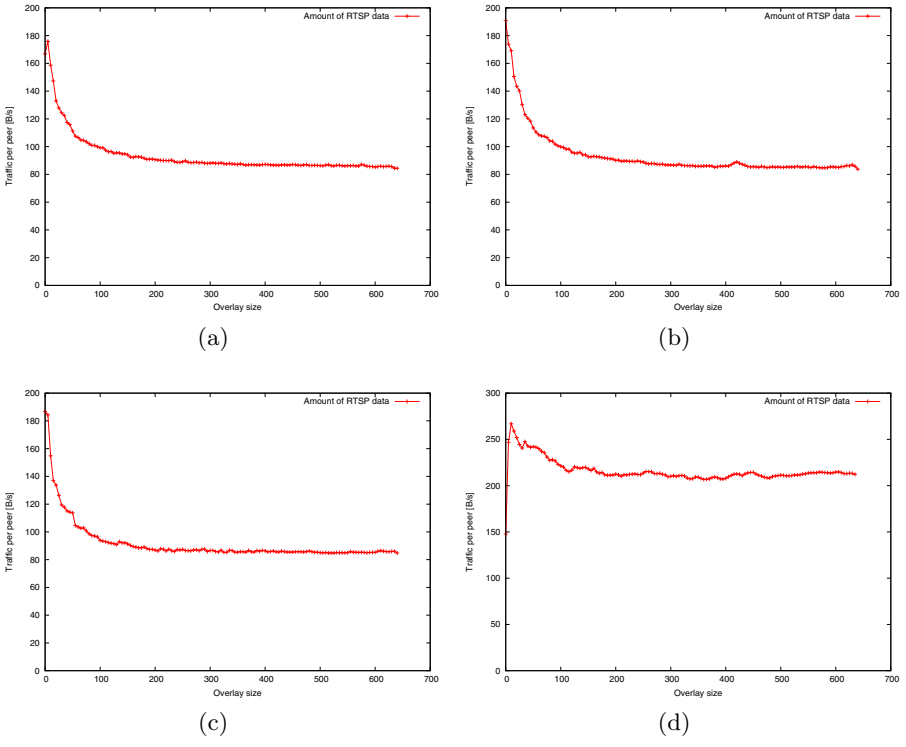

Fig. 9. The amount of sent RTSP data, (a) churning peers without packet loss recovery, (b) churning peers with packet loss recovery, (c) 1% uniformly distributed packet loss without loss recovery, and (d) 1% uniformly distributed packet loss with loss recovery

lost also with the packet loss recovery. It should be noted that the packet interval for both audio and video streams is circa 55 milliseconds, but the payload size and the total amount of lost bytes is larger for the video stream.

The amount of sent RTSP data in bytes per peer with and without RTSP-based packet loss recovery as a function of overlay size is shown in Fig. 9. From the figure we can see that with churning peers the values follow the same trend, but the packet loss recovery causes small overall increase to the signalling data. With a 1% uniformly distributed packet loss, the amount of RTSP data is more than doubled, but is still quite minimal compared to the combined bit rate 112kbps of the original RTP sessions.

7 Conclusions and Further Work

In a traditional streaming application based on RTP / RTCP, special mechanisms are available for signalling of packet losses and retransmission of lost packets. The traditional RTCP-based packet loss signalling is however insufficient in the peer-to-peer streaming case, since when a sender completely leaves

the network, it does not make sense to signal lost packets since they will never be received by the departed sender. This paper presents a scalable two-stage packet loss recovery mechanism for a real-time peer-to-peer streaming system using RTCP and RTSP.

Laboratory tests with the current RTSP-based packet loss recovery mechanism have shown that it is beneficial to try to ensure seamless media playback using packet retransmissions. Full implementation level support for the two-stage packet loss recovery mechanism is still needed to be able to verify the operation of the packet loss recovery mechanism and to be able to fine tune all parameters in the system, such as initial buffering time, partial RTP stream duration T_P and all waiting times in the two-stage packet recovery mechanism, to maximize the quality and minimize the data delivered in the network.

Mechanisms for handling packet losses using Forward Error Correction (FEC) in an RTSP-based P2P streaming system needs to be studied to find out the benefits and drawbacks of using FEC in addition to the current mechanisms based on packet retransmissions and peer replacement before the reception buffer underflows.

More advanced laboratory tests with different latencies, packet losses and throughputs between peers will highlight possible bottlenecks and usability issues in the system.

References

1. 3GPP: Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs. TS 26.234, 3rd Generation Partnership Project (3GPP) (March 2009)
2. Octoshape Homepage (March 2010), `http://www.octoshape.com/`
3. Ott, J., Wenger, S., Sato, N., Burmeister, C., Rey, J.: Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF). RFC 4585, Internet Engineering Task Force (July 2006)
4. Peltotalo, J., Bouazizi, I., Curcio, I., Hannuksela, M., Harju, J., Jantunen, A., Saukko, M., Väätämöinen, L.: Peer-to-Peer Streaming Technology Survey. In: Proceeding of the Seventh International Conference on Networking (ICN 2008), pp. 342–350 (April 2008) doi: 10.1109/ICN.2008.86
5. Peltotalo, J., Harju, J., Väätämöinen, L., Curcio, I., Bouazizi, I.: RTSP-based Mobile Peer-to-Peer Streaming System. International Journal of Digital Multimedia Broadcasting 2010, 15 pages (2010); Article ID 470813, doi:10.1155/2010/470813
6. Rey, J., Leon, D., Miyazaki, A., Varsa, V., Hakenberg, R.: RTP Retransmission Payload Format. RFC 4588, Internet Engineering Task Force (July 2006)
7. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force (July 2003)
8. Schulzrinne, H., Rao, A., Lanphier, R.: Real Time Streaming Protocol (RTSP). RFC 2326, Internet Engineering Task Force (April 1998)
9. Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., Stiemerling, M.: Real Time Streaming Protocol 2.0 (RTSP). Internet-Draft draft-ietf-mmusic-rfc2326bis-23, Internet Engineering Task Force (March 2010) (Work in progress)
10. SopCast Homepage (March 2010), `http://www.sopcast.org/`