Large-Scale Media Delivery using a Semi-Reliable Multicast Protocol*

Christiane M. Bortoleto, Lau Cheuk Lung
Pontifícia Univ. Católica do Paraná, Brazil
cbortoleto@ibest.com.br, lau@ppgia.pucpr.br

Frank Siqueira
Univ. Federal de Santa Catarina, Brazil
frank@inf.ufsc.br

Abstract

This paper proposes a semi-reliable multicast protocol that evaluates the necessity of retransmitting lost packets, taking into account the priority associated to each lost packet. The protocol relies on the neighboring receivers for retransmitting lost packets, resulting in much faster recovery. This paper also demonstrates how the proposed protocol is able to increase the quality of video streams transmitted over the Internet, employing knowledge on the transmitted MPEG frames to drive the recovery when the packet containing the frame is lost.

1. Introduction

The scientific literature is rich on examples of protocols which adopt a best-effort, multicast-based strategy for distributed multimedia applications, in which packets may be lost, corrupted or delivered out of order; and on reliable multicast protocols, which are employed by applications that demand fully reliable message delivery (e.g., fault tolerant applications). Semi-reliable multicast, a more recent proposal that was not fully explored yet [3,4,5,6,7], emerges as an alternative for these two approaches, which is appropriate for applications such as audio and video streaming.

This paper presents an efficient semi-reliable multicast protocol, designed for distributing information with different reliability requirements, which are taken into account by the data recovery mechanisms implemented by the protocol. This paper also shows how this protocol can be employed by group-based distributed multimedia applications (e.g. digital video multicast) for efficient delivery of media streams. The protocol requires that the video encoders employed by the applications define some kind of frame hierarchy, allowing the establishment of priorities that are taken into account while recovering lost frames.

The remainder of this paper is organized as follows. Section 2 defines the main concepts related to semi-reliable multicast. A recovery algorithm for semi-reliable multicast is described in section 3. The implementation of a semi-reliable protocol for delivery of MPEG video streams which employs the proposed recovery algorithm is described in section 4. An application scenario and a performance evaluation study are presented in section 5. Finally, the conclusions of the authors are presented in section 6, together with perspectives for further developments.

2. Semi-Reliable Multicast

Semi-reliable multicast is a communication paradigm in which not every packet is necessarily retransmitted when it is lost; most important packets for the application are given higher priority. This means that the reliable delivery for a given set of packets which will be sent in a receiver group is granted only to a subset of these (i.e., the packets with higher priority). The other packets that suffered transmission errors, which have lower priority, will only be corrected if the network conditions (i.e. traffic, congestion, latency, and so on) allow it.

It is possible to find in the technical literature papers proposing multicast protocols with semi-reliable characteristics. PRTP (Partially Reliable Transport Protocol) [13], for example, is a partially reliable protocol that does not insist on recovering all the lost data. Instead, it recovers only part of the lost packets, based on a minimum reliability defined by the

* This work is partially supported by CNPq (Brazilian National Research Council) through processes 481523/2004-9 and 506639/2004-5.

application. When the parameter level of reliability is above the requested limit, the receiver that detects a lost packet does not request its retransmission.

PRMP (Polling-based Reliable Multicast Protocol) [1] is a reliable multicast protocol with a source-based recovery mechanism, in which the packet rate at the source is controlled in a way that it does not exceed source or network capacity.

SRP [12] (Selective Retransmission Protocol) adjusts the amount of retransmission based on QoS factors, including total loss, latency, RTT (round trip time), network congestion and user requirements. The protocol is able to adjust the loss and latency levels of each application, employing a decision algorithm to determine if a retransmission request for a lost packet should be answered or not. By adopting this strategy, just a fraction of the lost data is retransmitted.

The WAIT protocol [9] provides an improved quality of service for applications exchanging data through the Internet, adjusting itself to different quality requirements and reducing the network load.

A different approach, which is based on semantically reliable multicast protocols, is presented in [11]. The proposed approach eliminates obsolete messages to sustain a higher throughput and to avoid network congestion. Both the source and the receivers eliminate obsolete messages from their buffers when the buffer capacity is reached.

Interesting strategies for packet recovery are also found in reliable multicast protocols. SRM (Scalable Reliable Multicast) [6], for example, is a reliable multicast protocol in which the retransmission is performed by the receivers. In this protocol, every time a loss is detected, a negative acknowledgement (NACK) is sent to the whole multicast group and any member having the requested packet may resend it. To avoid packet flooding, every receiver waits for a random time before sending a NACK and before retransmitting a lost packet. If another node multicasts the packet or NACK it was about to send, it assumes that it is not necessary to do it.

These protocols present important improvements necessary for some applications; however, they adopt a common strategy for all transmitted packets. In the next sections we propose a semi-reliable multicast protocol and a recovery algorithm that take into account the reliability of different classes of packets, which can be adjusted for different application scenarios. In this paper we also describe how this protocol may be employed for multimedia delivery, and show that it is able to improve the quality of the exhibited media.

3. A Recovery Algorithm for Semi-Reliable Multicast

The Semi-Reliable Recovery Algorithm proposed in this paper is based on two main principles:

- Error correction can be made not only by the source, but also by the receivers belonging to a multicast group;
- Error correction mechanisms must take into account information on the lost data in order to, based on quality of service (QoS) requirements, decide if it is necessary to recover this data.

Packets are multicast directly to the group, without requiring previous knowledge of its members. Packet loss is determined by finding gaps in the sequence numbers of packets belonging to a packet flow. In order to decide if a lost packet must be retransmitted, a receiver must know the priority of this packet. This information must be either carried by control messages or by previous or subsequent packets in the same packet flow. Therefore, when a packet is lost, the application can still determine if it must be recovered.

The proposed algorithm is described in Figure 1. Each receiver stores the new received packets in its buffer (line 4), from where the application will fetch them. In line 6, the receiver cancels any possible retransmission request of the received message. If the received message has already been received, it is a retransmission sent to another receiver that requested this packet, and it is not necessary to retransmit it again (line 9). Then, receivers detect lost packets (line 11) through the search for gaps in the buffer, which are detected verifying the sequence number of stored packets. When a loss is detected, the receivers evaluate if it is necessary to request retransmission through a NACK message (lines 12 and 13). This evaluation is based on the current packet loss rate and on application-specific QoS requirements, which represent the relevance of the lost packet for the receiver and the usability of the lost packet by the time it arrived at the receiver. Network parameters taken into account are packet loss rate and acceptable delay. A lost packet that is considered relevant (line 14) has its retransmission requested through a NACK message that is multicast to the group after waiting for a random time (lines 15 to 19). Otherwise, if the packet is not relevant, the receiver ignores the lost packet. Any process (sender or receiver) which receives a NACK and has the requested packet, evaluates again the parameters (line 26) and, if retransmission is required, multicasts it again to the group after a random time (lines 28 to 32).

```
RECEIVER ALGORITHM
1.     WHEN receive(m)                    {wait until a message is received}
2.        IF m IS data                    {if this is a data message}
3.           IF m ∉ buffer                {if m is not in the buffer}
4.              buffer ← m                {add m to buffer}
5.              IF m ∈ nack_list          {if there is a NACK pending for this message}
6.                 nack_list → m          {cancel NACK for m}
7.              END-IF
8.           ELSE IF m ∈ recovery_list    {if recovery of m was requested by another receiver}
9.              recovery_list → m         {cancels retransmission of m}
10.          END-IF
11.          FOR EVERY ml ∉ buffer        {for every message delayed or lost}
12.             IF ml.expired = false AND {if the message has not expired and the required...}
13.                ml.loss > current_loss {...loss rate for ml is greater than the current loss}
14.                nack_list ← ml         {add ml to the list of messages to be recovered}
15.                WAIT random(T_nack)    {wait for a random time}
16.                IF ml ∈ nack_list      {if ml has not been received or NACKed by others}
17.                   multicast(ml)       {multicast NACK for this message to the group}
18.                   nack_list → ml      {remove from the list of NACKs to be sent}
19.                END-IF
20.             END-FOR
21.          END-IF
22.       ELSE IF m IS nack               {if this message is a retransmission request}
23.          IF m ∈ nack_list             {if the retransmission of m was waiting to be requested}
24.             nack_list → m             {cancel NACK for m because it has already been sent}
25.          ELSE IF m ∈ buffer           {if the requested message is in the buffer}
26.             IF m.loss > current_loss  {if m must be recovered to fulfill the required quality}
27.                recovery_list ← m      {add m to the list of messages to be recovered}
28.                WAIT random(T_repair)  {wait for a random time}
29.                IF m ∈ recovery_list   {if m was not recovered during this time}
30.                   multicast(m)        {multicast the lost message to the group}
31.                   recovery_list → m   {remove m from the recovery list}
32.                END-IF
33.             END-IF
34.          END-IF
35.       END-IF
36.    END-WHEN
```

Figure 1. Proposed Protocol – Receiver Algorithm

The loss rate at each receiver is calculated based on a sample window of size N, i.e., the last N multicast packets. The current_loss variable (lines 13 and 26) holds the percentage of lost packets in the sample, calculated based on the number of missing packets in the sample and the sample size. The relevance of a packet is verified through the reception buffer. The expired attribute of buffered packets (line 12) is set to true when the application tries to fetch a packet from the buffer and it is not available.

To avoid NACK or retransmission explosions, a wait function interrupts the execution during a random interval. Before sending a NACK, a receiver R_i waits for a random time limited by T_{nack} (line 15). If during this time it receives a NACK from another receiver requesting the same packet, the receiver cancels its NACK (line 24). In a similar way, when it receives a NACK and has the requested packet, it waits for a random time limited by T_{repair} before multicasting this packet (line 28). However, if within this period it receives the requested packet, the receiver cancels the retransmission (line 9).

4. A Semi-Reliable Multicast Protocol for Media Transmission

Multimedia data is generated with a fixed rate, and frames have to be received and rendered in the receiver with a similar rate to keep the original meaning of the media. Thus, each packet has a deadline associated to it. Besides, the loss ratio (i.e., the rate of data packets lost during network transmission or delivered after the time they should have been rendered) should also be within boundaries defined by the application. So, the definition of QoS requirements includes finding the acceptable boundaries for transmission errors and jitter.

A traditional reliable multicast mechanism is not appropriated for multimedia multicast for many reasons. The retransmission strategy with timeout achieves reliability but implies in latency increase. Multimedia applications can tolerate errors due to lost and corrupted frames as long as the loss rate is within an acceptable limit. Thus, a multimedia transport protocol demands semi-reliable delivery, where

respecting deadlines is more relevant than delivering every single media packet.

4.1. The MPEG-2 Standard

The MPEG-2 Motion Picture Experts Group) compression algorithm [1] is based on pixel correlation and translational movement correlation between consecutive frames. Most frames in an image sequence are very similar, except for differences due to movement, so it is possible to encode a frame calculating the movement vector related to the previous frame [6].

The MPEG-2 standard defines three types of frames:

- I Frames (Intra-coded): a full image, like a JPG picture;
- P Frames (Predictive): frames defined based on the previous I frame;
- B Frames (Bidirectional): frames defined using the previous and the next I or P frame.

A sequence starting with an I frame and ending in the next I frame is called a group of pictures (GOP). The dependencies between frames in a GOP are illustrated by Figure 2.

I frames do not require other frames to be decoded, but they are necessary for decoding P and B frames. If an I frame is lost during transmission, it will not be possible to decode the following frames that arrive before the next I frame – i.e., the whole GOP. P frames are needed for decoding B frames and are based on forward prediction using the previous frame as reference, which can be a P or I frame. If a P frame is lost, all the previous B frames until the last P frame and the subsequent frames in the GOP cannot be decoded. B frames, though, are not necessary for decoding other frames.

4.2. Design Principles

This protocol uses the frame hierarchy defined by the MPEG standard for classifying MPEG frames and deciding when a lost frame should be recovered, enforcing a semi-reliable multicast policy.

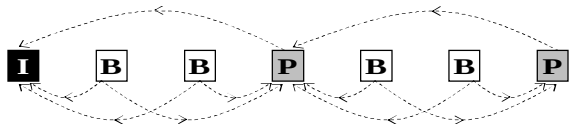

Figure 2. Example of MPEG-2 Group of Pictures (GOP)

Once the frames are classified, error correction can be performed by the sender or the receivers, according to the lost frame type (I, P or B) and to the network conditions.

The main principle behind the protocol is that every I frame is subject to recovery while it is still valid; on the other hand, P and B frames are retransmitted only if the transmission environment conditions allow it. Both P and B frames are associated to loss rates, being retransmitted only if the current loss rate does not exceed the loss rate specified by this frame type. In general, B frames are considered less important for the quality of the rendered video than P frames, therefore they are allowed to suffer greater loss.

4.3. Protocol Implementation

A prototype of this protocol has been implemented in Java in order to have its behavior evaluated. IP multicast was employed as the network protocol, with the recovery mechanism described in session 3 implemented as a layer on top of it. At this stage of the development, flow control mechanisms have not been implemented.

A double buffer is employed to store received frames before they are rendered by the application. While the network fills the buffer with new frames, the stored frames are being read from the buffer by the application in order to be rendered. The protocol stores frames that have already been rendered for one second, if memory is available, so that they can be retransmitted if required by another receiver. In this case, the application will employ the algorithm describe in session 3 in order to decide if it will answer the retransmission request.

Figure 3 illustrates the buffer structure. Each column corresponds to a GOP. Each frame is associated to a frame type and a sequence number. Shaded frames have already been rendered, and are kept in the buffer for retransmission purposes. Frames were marked as 'F_i' were lost or delayed, preventing their exhibition and, depending on the frame type, the exhibition of frames that required them for being rendered (e.g., frames 25 to 27 due to the loss of frame P_{24}, and frames 36 to 41 due to the loss of frame I_{35}). Frames from 42 to 65 have not reached the time for being rendered, and the missing ones may still be recovered in time.

	Expired						Buffered			
	I_{00}	I_{07}	I_{14}	I_{21}	I_{28}	I_{35}	I_{42}	I_{49}	I_{56}	I_{63}
	B_{01}	B_{08}	B_{15}	B_{22}	B_{29}	B_{36}	B_{43}	B_{50}	B_{57}	B_{64}
G	B_{02}	B_{09}	B_{16}	B_{23}	B_{30}	B_{37}	B_{44}	B_{51}	B_{58}	B_{65}
O	P_{03}	P_{10}	P_{17}	P_{24}	P_{31}	P_{38}	P_{45}	P_{52}	P_{59}	
P	B_{04}	B_{11}	B_{18}	B_{25}	B_{32}	B_{39}	B_{46}	B_{53}	B_{60}	
	B_{05}	B_{12}	B_{19}	B_{26}	B_{33}	B_{40}	B_{47}	B_{54}	B_{61}	
	P_{06}	P_{13}	P_{20}	P_{27}	P_{34}	P_{41}	P_{48}	P_{55}	P_{62}	

F_i	Type ‘F’ frame, seq. no. ‘i’, received
F_i	Type ‘F’ frame, seq. no. ‘i’, received and rendered
F_i	Type ‘F’ frame, seq. no. ‘i’, lost or delayed

Figure 3. Packet Buffer

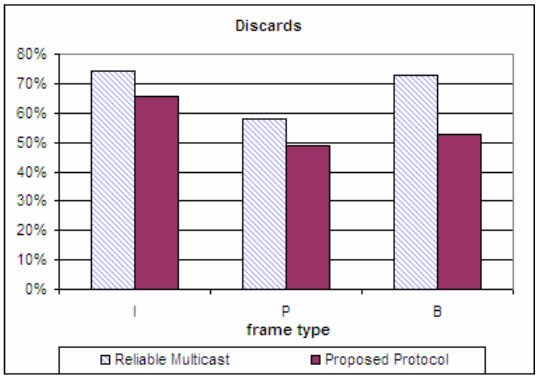

Figure 4. Discarded Packets

For simplicity, we assume that a single video frame is multicast by the video source in a datagram. The protocol header specifies the sequence number of this frame, and the frame type – I, P or B – of this and the last N-1 frames, where N is the size of the GOP. Therefore, it is possible to know what kind of frame is missing when there is a gap in the sequence number of received frames.

A frame is marked as expired in the corresponding position of the buffer if the application tries to read it and it has not been received (i.e., it was either lost or delayed by the network) or another frame that is necessary to render it is missing. After this moment, it is useless to try to recover this frame, since it missed the time to be exhibited.

The loss rates associated to P and B frames can be adjusted so that the user can identify the values that provide the best video quality in a particular application scenario.

5. Application Scenario

In order to evaluate the proposed protocol, we have simulated its behavior using the Simmcast [1] network simulator.

In the performed tests, the proposed protocol, a plain multicast protocol, a multicast protocol with retransmission and a reliable multicast protocol were compared in terms of error correction, recovery time, receiver overload and video quality factor [5].

The network topology employed during these tests was carefully chosen in order to evaluate the adequacy of these protocols for multicast delivery in a large-scale network. The chosen topology allowed the evaluation of very similar conditions to those existing on the Internet, with the distance from the source to the receiver ranging from 1 to 15 hops.

5.1. Performance Measurements

Experiments were made in order to establish boundaries for the loss rate parameters for different frame types. The performance of the protocol was evaluated in terms of error correction, delay and a mathematical relation between recovered frames and their importance for the quality of the rendered media. The chosen experimental values were 83% of loss for P-frames and 50% for B-frames.

Simulations were performed using the same conditions for all four protocols. In terms of lost packets recovery, the proposed protocol has presented an average of 81.7 % of recovery against 2% of multicast with NACK (simple retransmission) and 89% of the reliable multicast protocol. The best performance from the proposed protocol and the reliable multicast can be credited to the receiver-based retransmission (about 90% of the recovered packets came from other receivers instead of the source). The reliable multicast has achieved a better rate due to its attempts to recover every lost packet, no matter the temporal relevance of the packet for the application.

The reliable multicast protocol recovers more frames than the proposed protocol, but great part of recovered data is discarded – 74% of recovered I-frames against an average of 66% in the proposed protocol, as shown by Figure 4. It means that the proposed protocol is more efficient, once the recovered packets are more often useful for the application by the time they arrive.

An important result obtained with the proposed protocol was the NACK suppression: an average of 31,5%. The results show that the rate increases when the receiver is more distant from the source (the average is 55% for the most distant ones), since receivers close to the source detect losses and send NACKs more quickly than the more distant receivers.

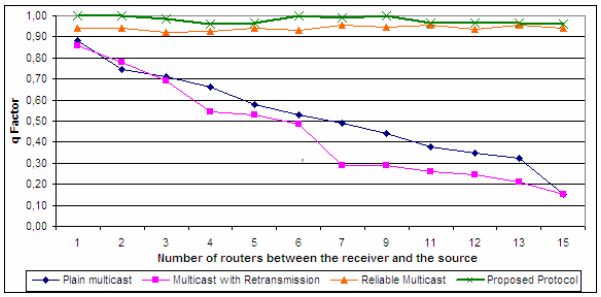

Figure 5. Video Quality Factor

Comparing the recovering times, the proposed protocol was more efficient: its average was 214 ms against 299 ms of the multicast with retransmission and 368 ms of the reliable multicast.

Another parameter used in the evaluation was the Video Quality Factor (q), defined in [5]. This parameter consists in a metric based on the GOP structure to evaluate video quality. The formula takes into account direct losses (the ones caused by losing the frame itself) and indirect losses (caused by the loss of another frame). It is important to highlight that this metric claims to evaluate transport information regarding the video flow, and not the video quality perceived by the user.

As shown in Figure 5, the proposed protocol has presented a better performance compared to all the other tested protocols, despite not recovering as many frames as the reliable multicast protocol. This result was obtained due to the selective discard of packets containing less relevant frames.

6. Conclusions

This paper presented a semi-reliable multicast protocol which is able to improve the quality of the media delivery to a group through the network. Based on the MPEG standard and on the multicast technology, this protocol provides semi-reliable delivery of video frames, improving the quality of media multicast through the Internet.

An alternative approach, based on a different technique, was proposed in the literature. Yavatkar and Manoj have proposed a quasi-reliable multicast transport protocol for transmitting multimedia information in large scale [14]. The authors state that, due to the nature of multimedia communication, the protocol must use forward error correction to avoid delays inherent to flow-based and error control techniques. Our approach is able to deliver media with quality close to optimal without imposing the overhead caused by forward error correction techniques.

The proposed semi-reliable multicast protocol is still under development. In the near future we intend to improve the protocol by adding congestion and flow control mechanisms. We also intend to integrate adaptation mechanisms to the protocol, which will allow the automatic adjustment of the parameters considered by the algorithm based on current network conditions.

7. References

[1] M. Barcellos et al., “Simmcast: a Simulation Tool for Multicast Protocol Evaluation”. *XI Simpósio Brasileiro de Redes de Computadores*, Florianópolis, 2001.

[2] L. Chiariglione, “Short MPEG-2 Description”, April 2000. http://mpeg.telecomitalialab.com/standards/mpeg-2/mpeg-2.htm.

[3] S. Floyd et al., “A reliable multicast framework for light-weight sessions and application level framing”. *Proc. of the ACM SIGCOMM 95*, Aug. 1995, pp. 345-356.

[4] P. Mane, “WAIT: Selective Loss Recovery for Multimedia Multicast”. M.Sc. Thesis, Computer Science Department, WPI, 2000.

[5] R.F. Martins, C.A. Leite, J.-M. Farines, “Toward Quality Evaluation and Improvement of a MPEG Vídeo Stream. *Proc. of the 3th IEEE Latin American Network Operations and Management Symposium – LANOMS’03*, Foz do Iguaçu, Brazil, 2003.

[6] J. Pereira et al., “Semantically Reliable Multicast Definition, Implementation and Performance Evaluation”. *IEEE Transactions on Computers*, vol. 52 no.2, 2003, pp. 150-165.

[7] M. Piecuch et al., “A Selective Retransmission Protocol for Multimedia on the Internet”. *Proceedings of SPIE International Symposium on Multimedia Systems and Applications*. Nov. 2000.

[8] S. Schneyer et al., “PRTP: A Partially Reliable Transport Protocol for Multimedia Applications”. *Proceedings of ISIMADE*, Baden-Baden, Germany, Aug. 1999.

[9] X. Xu et al., “Resilient Multicast Support for Continuous-Media Applications”. *Proceedings of the NOSSDAV’97*, 1997.

[10] R. Yavatkar, L. Manoj, “Optimistic Strategies for Large-Scale Dissemination of Multimedia Information”, *First ACM Int. Conf. on Multimedia*, 1993.