Network Working Group
Request for Comments: 2354
Category: Informational

C. Perkins
O. Hodson
University College London

June 1998

Options for Repair of Streaming Media

Status of this Memo

Copyright Notice

Abstract

This document summarizes a range of possible techniques for the repair of continuous media streams subject to packet loss. The techniques discussed include redundant transmission, retransmission, interleaving and forward error correction. The range of applicability of these techniques is noted, together with the protocol requirements and dependencies.

1 Introduction

A number of applications have emerged which use RTP/UDP transport to deliver continuous media streams. Due to the unreliable nature of UDP packet delivery, the quality of the received stream will be adversely affected by packet loss. A number of techniques exist by which the effects of packet loss may be repaired. These techniques have a wide range of applicability and require varying degrees of protocol support. In this document, a number of such techniques are discussed, and recommendations for their applicability made.

It should be noted that this document is introductory in nature, and does not attempt to be comprehensive. In particular, we restrict our discussion to repair techniques which require the involvement of the sender of a media stream, and do not discuss possibilities for receiver based repair.

For a more detailed survey, the reader is referred to [5].

2 Terminology and Protocol Framework

A unit is defined to be a timed interval of media data, typically derived from the workings of the media coder. A packet comprises one or more units, encapsulated for transmission over the network. For example, many audio coders operate on 20ms units, which are typically combined to produce 40ms or 80ms packets for transmission. The framework of RTP [18] is assumed. This implies that packets have a sequence number and timestamp. The sequence number denotes the order in which packets are transmitted, and is used to detect losses. The timestamp is used to determine the playout order of units. Most loss recovery schemes rely on units being sent out of order, so an application must use the RTP timestamp to schedule playout.

The use of RTP allows for several different media coders, with a payload type field being used to distinguish between these at the receiver. Some loss repair schemes send multiple copies of units, at different times and possibly with different encodings, to increase the probability that a receiver has something to decode. A receiver is assumed to have a ‘quality’ ranking of the differing encodings, and so is capable of choosing the ‘best’ unit for playout, given multiple options.

A session is defined as interactive if the end-to-end delay is less then 250ms, including media coding and decoding, network transit and host buffering.

3 Network Loss Characteristics

If it is desired to repair a media stream subject to packet loss, it is useful to have some knowledge of the loss characteristics which are likely to be encountered. A number of studies have been conducted on the loss characteristics of the Mbone [2, 8, 21] and although the results vary somewhat, the broad conclusion is clear: in a large conference it is inevitable that some receivers will experience packet loss. Packet traces taken by Handley [8] show a session in which most receivers experience loss in the range 2-5%, with a somewhat smaller number seeing significantly higher loss rates. Other studies have presented broadly similar results.

It has also been shown that the vast majority of losses are of single packets. Burst losses of two or more packets are around an order of magnitude less frequent than single packet loss, although they do occur more often than would be expected from a purely random process. Longer burst losses (of the order of tens of packets) occur infrequently. These results are consistent with a network where small amounts of transient congestion cause the majority of packet loss. In a few cases, a network link is found to be severely

overloaded, and large amount of loss results.

The primary focus of a repair scheme must, therefore, be to correct single packet loss, since this is by far the most frequent occurrence. It is desirable that losses of a relatively small number of consecutive packets may also be repaired, since such losses represent a small but noticeable fraction of observed losses. The correction of large bursts of loss is of considerably less importance.

4 Loss Mitigation Schemes

In the following sections, four loss mitigation schemes are discussed. These schemes have been discussed in the literature a number of times, and found to be of use in a number of scenarios. Each technique is briefly described, and its advantages and disadvantages noted.

4.1 Retransmission

Retransmission of lost packets is an obvious means by which loss may be repaired. It is clearly of value in non-interactive applications, with relaxed delay bounds, but the delay imposed means that it does not typically perform well for interactive use.

In addition to the possibly high latency, there is a potentially large bandwidth overhead to the use of retransmission. Not only are units of data sent multiple times, but additional control traffic must flow to request the retransmission. It has been shown that, in a large Mbone session, most packets are lost by at least one receiver [8]. In this case the overhead of requesting retransmission for most packets may be such that the use of forward error correction is more acceptable. This leads to a natural synergy between the two mechanisms, with a forward error correction scheme being used to repair all single packet losses, and those receivers experiencing burst losses, and willing to accept the additional latency, using retransmission based repair as an additional recovery mechanism. Similar mechanisms have been used in a number of reliable multicast schemes, and have received some discussion in the literature [9, 13].

In order to reduce the overhead of retransmission, the retransmitted units may be piggy-backed onto the ongoing transmission, using a payload format such as that described in [15]. This also allows for the retransmission to be recoded in a different format, to further reduce the bandwidth overhead. As an alternative, FEC information may be sent in response to retransmission requests [13], allowing a single retransmission to potentially repair several losses. The choice of a retransmission request algorithm which is both timely and

network friendly is an area of current study. An obvious starting point is the SRM protocol [7], and experiments have been conducted using this, and with a low-delay variant, STORM [20]. This work shows the trade-off between latency and quality for retransmission based repair schemes, and illustrates that retransmission is an effective approach to repair for applications which can tolerate the latency.

There is no standard protocol framework for requesting retransmission of streaming media. An experimental RTP profile extension for SRM-style retransmission requests has described in [14].

4.2 Forward Error Correction

Forward error correction (FEC) is the means by which repair data is added to a media stream, such that packet loss can be repaired by the receiver of that stream with no further reference to the sender. There are two classes of repair data which may be added to a stream: those which are independent of the contents of the stream, and those which use knowledge of the stream to improve the repair process.

4.2.1 Media-Independent FEC

A number of media-independent FEC schemes have been proposed for use with streamed media. These techniques add redundant data, which is transmitted in separate packets, to a media stream. Traditionally, FEC techniques are described as loss detecting and/or loss correcting. In the case of streamed media, loss detection is provided by the sequence numbers in RTP packets.

The redundant FEC data is typically calculated using the mathematics of finite fields [1]. The simplest of finite field is GF(2) where addition is just the eXclusive-OR operation. Basic FEC schemes transmit k data packets with n-k parity packets allowing the reconstruction of the original data from any k of the n transmitted packets. Budge et al [4] proposed applying the XOR operation across different combinations of the media data with the redundant data transmitted separately as parity packets. These vary the pattern of packets over which the parity is calculated, and hence have different bandwidth, latency and loss repair characteristics.

Parity-based FEC based techniques have a significant advantage in that they are media independent, and provide exact repair for lost packets. In addition, the processing requirements are relatively light, especially when compared with some media-specific FEC (redundancy) schemes which use very low bandwidth, but high complexity encodings. The disadvantage of parity based FEC is that the codings have higher latency in comparison with the media-specific

schemes discussed in following section.

A number of FEC schemes exist which are based on higher-order finite fields, for example Reed-Solomon (RS) codes, which are more sophisticated and computationally demanding. These are usually structured so that they have good burst loss protection, although this typically comes at the expense of increased latency. Dependent on the observed loss patterns, such codes may give improved performance, compared to parity-based FEC.

An RTP payload format for generic FEC, suitable for both parity based and Reed-Solomon encoded FEC is defined in [17].

4.2.2 Media-Specific FEC

The basis of media-specific FEC is to employ knowledge of a media compression scheme to achieve more efficient repair of a stream than can otherwise be achieved. To repair a stream subject to packet loss, it is necessary to add redundancy to that stream: some information is added which is not required in the absence of packet loss, but which can be used to recover from that loss.

The nature of a media stream affects the means by which the redundancy is added. If units of media data are packets, or if multiple units are included in a packet, it is logical to use the unit as the level of redundancy, and to send duplicate units. By recoding the redundant copy of a unit, significant bandwidth savings may be made, at the expense of additional computational complexity and approximate repair. This approach has been advocated for use with streaming audio [2, 10] and has been shown to perform well. An RTP payload format for this form of redundancy has been defined [15].

If media units span multiple packets, for instance video, it is sensible to include redundancy directly within the output of a codec. For example the proposed RTP payload for H.263+ [3] includes multiple copies of key portions of the stream, separated to avoid the problems of packet loss. The advantages of this second approach is efficiency: the codec designer knows exactly which portions of the stream are most important to protect, and low complexity since each unit is coded once only.

An alternative approach is to apply media-independent FEC techniques to the most significant bits of a codecs output, rather than applying it over the entire packet. Several codec descriptions include bit sensitivities that make this feasible. This approach has low computational cost and can be tailored to represent an arbitrary fraction of the transmitted data.

The use of media-specific FEC has the advantage of low-latency, with only a single-packet delay being added. This makes it suitable for interactive applications, where large end-to-end delays cannot be tolerated. In a uni-directional non-interactive environment it is possible to delay sending the redundant data, achieving improved performance in the presence of burst losses [11], at the expense of additional latency.

4.3 Interleaving

When the unit size is smaller than the packet size, and end-to-end delay is unimportant, interleaving [16] is a useful technique for reducing the effects of loss. Units are resequenced before transmission, so that originally adjacent units are separated by a guaranteed distance in the transmitted stream, and returned to their original order at the receiver. Interleaving disperses the effect of packet losses. If, for example, units are 5ms in length and packets 20ms (ie: 4 units per packet), then the first packet could contain units 1, 5, 9, 13; the second packet would contain units 2, 6, 10, 14; and so on. It can be seen that the loss of a single packet from an interleaved stream results in multiple small gaps in the reconstructed stream, as opposed to the single large gap which would occur in a non-interleaved stream. In many cases it is easier to reconstruct a stream with such loss patterns, although this is clearly media and codec dependent. Note that the size of the gaps is dependent on the degree of interleaving used, and can be made arbitrarily small at the expense of additional latency.

The obvious disadvantage of interleaving is that it increases latency. This limits the use of this technique for interactive applications, although it performs well for non-interactive use. The major advantage of interleaving is that it does not increase the bandwidth requirements of a stream.

A potential RTP payload format for interleaved data is a simple extension of the redundant audio payload [15]. That payload requires that the redundant copy of a unit is sent after the primary. If this restriction is removed, it is possible to transmit an arbitrary interleaving of units with this payload format.

5 Recommendations

If the desired scenario is a non-interactive uni-directional transmission, in the style of a radio or television broadcast, latency is of considerably less importance than reception quality. In this case, the use of interleaving, retransmission based repair or FEC is appropriate. If approximate repair is acceptable, interleaving is clearly to be preferred, since it does not increase

the bandwidth of a stream. Media independent FEC is typically the next best option, since a single FEC packet has the potential to repair multiple lost packets, providing efficient transmission.

In an interactive session, the delay imposed by the use of interleaving and retransmission is not acceptable, and a low-latency FEC scheme is the only means of repair suitable. The choice between media independent and media specific forward error correction is less clear-cut: media-specific FEC can be made more efficient, but requires modification to the output of the codec. When defining the packet format for a new codec, this is clearly an appropriate technique, and should be encouraged.

If an existing codec is to be used, a media independent forward error correction scheme is usually easier to implement, and can perform well. A media stream protected in this way may be augmented with retransmission based repair with minimal overhead, providing improved quality for those receivers willing to tolerate additional delay, and allowing interactivity for those receivers which desire it. Whilst the addition of FEC data to an media stream is an effective means by which that stream may be protected against packet loss, application designers should be aware that the addition of large amounts of repair data when loss is detected will increase network congestion, and hence packet loss, leading to a worsening of the problem which the use of error correction coding was intended to solve.

At the time of writing, there is no standard solution to the problem of congestion control for streamed media which can be used to solve this problem. There have, however, been a number of contributions which show the likely form the solution will take [12, 19]. This work typically used some form of layered encoding of data over multiple channels, with receivers joining and leaving layers in response to packet-loss (which indicates congestion). The aim of such schemes is to emulate the congestion control behavior of a TCP stream, and hence compete fairly with non-real time traffic. This is necessary for stable network behavior in the presence of much streamed media.

Since streaming media applications are in use now, without congestion control, it is important to give some advice to authors of those tools as to the behavior which is acceptable, until congestion control mechanisms can be deployed. The remainder of this section uses the throughput of a TCP connection over a link with a given loss rate as an example to indicate behavior which may be classified as reasonable.

As a number of authors have noted (eg: [6]), the loss rate and throughput of a TCP connection are approximately related as follows:

$$T = (s * c) / (RTT * \sqrt{p})$$

where T is the observed throughput in octets per second, s is the packet size in octets, RTT is the round-trip time for the session in seconds, p is the packet loss rate and c is a constant in the range 0.9...1.5 (a value of 1.22 has been suggested [6]). Using this relation, one may determine the packet loss rate which would result in a given throughput for a particular session, if a TCP connection was used.

Whilst this relation between packet loss rate and throughput is specific to the TCP congestion control algorithm, it also provides an estimate of the acceptable loss rate for a streaming media application using the same network path, which wishes to coexist fairly with TCP traffic. Clearly this is not sufficient for fair sharing of a link with TCP traffic, since it does not capture the dynamic behavior of the connection, merely the average behavior, but it does provide one definition of "reasonable" behavior in the absence of real congestion control.

For example, an RTP audio session with DVI encoding and 40ms data packets will have 40 bytes RTP/UDP/IP header, 4 bytes codec state and 160 bytes of media data, giving a packet size, s, of 204 bytes. It will send 25 packets per second, giving T = 4800. It is possible to estimate the round-trip time from RTCP reception report statistics (say 200 milliseconds for the purpose of example). Substituting these values into the above equation, we estimate a "reasonable" packet loss rate, p, of 6.7%. This would represent an upper bound on the packet loss rate which this application should be designed to tolerate.

It should be noted that a round trip time estimate based on RTCP reception report statistics is, at best, approximate; and that a round trip time for a multicast group can only be an ‘average’ measure. This implies that the TCP equivalent throughput/loss rate determined by this relation will be an approximation of the upper bound to the rate a TCP connection would actually achieve.

6 Security Considerations

Some of the repair techniques discussed in this document result in the transmission of additional traffic to correct for the effects of packet loss. Application designers should be aware that the transmission of large amounts of repair traffic will increase network congestion, and hence packet loss, leading to a worsening of the problem which the use of error correction was intended to solve. At its worst, this can lead to excessive network congestion and may constitute a denial of service attack. Section 5 discusses this in

more detail, and provides guidelines for prevention of this problem.

7 Summary

Streaming media applications using the Internet will be subject to packet loss due to the unreliable nature of UDP packet delivery. This document has summarized the typical loss patterns seen on the public Mbone at the time of writing, and a range of techniques for recovery from such losses. We have further discussed the need for congestion control, and provided some guidelines as to reasonable behavior for streaming applications in the interim until congestion control can be deployed.

8 Acknowledgments

The authors wish to thank Phil Karn and Lorenzo Vicisano for their helpful comments.

9 Authors' Addresses

Colin Perkins
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom

EMail: c.perkins@cs.ucl.ac.uk

Orion Hodson
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom

EMail: o.hodson@cs.ucl.ac.uk

References

[1] R.E. Blahut. Theory and Practice ofError Control Codes. Addison Wesley, 1983.

[2] J.-C. Bolot and A. Vega-Garcia. The case for FEC based error control for packet audio in the Internet. To appear in ACM Multimedia Systems.

[3] C. Bormann, L. Cline, G. Deisher, T. Gardos, C. Maciocco, D. Newell, J. Ott, S. Wenger, and C. Zhu. RTP payload format for the 1998 version of ITU-T reccomendation H.263 video (H.263+). Work in Progress.

[4] D. Budge, R. McKenzie, W. Mills, W. Diss, and P. Long. Media-independent error correction using RTP, Work in Progress.

[5] G. Carle and E. W. Biersack. Survey of error recovery techniques for IP-based audio-visual multicast applications. IEEE Network, 11(6):24--36, November/December 1997.

[6] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. Submitted to IEEE/ACM Transactions on Networking, February 1998.

[7] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and applications level framing. IEEE/ACM Transactions on Networking, 1995.

[8] M. Handley. An examination of Mbone performance. USC/ISI Research Report: ISI/RR-97-450, April 1997.

[9] M. Handley and J. Crowcroft. Network text editor (NTE): A scalable shared text editor for the Mbone. In Proceedings ACM SIGCOMM'97, Cannes, France, September 1997.

[10] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. In Proceedings of INET'95, 1995.

[11] I. Kouvelas, O. Hodson, V. Hardman, and J. Crowcroft. Redundancy control in real-time Internet audio conferencing. In Proceedings of AVSPN'97, Aberdeen, Scotland, September 1997.

[12] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In Proceedings ACM SIGCOMM'96, Stanford, CA., August 1996.

[13] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. In Proceedings ACM SIGCOMM'97, Cannes, France, September 1997.

[14] P. Parnes. RTP extension for scalable reliable multicast, Work in Progress.

[15] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J-C., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[16] J.L. Ramsey. Realization of optimum interleavers. IEEE Transactions on Information Theory, IT-16:338--345, May 1970.

[17] J. Rosenberg and H. Schulzrinne. An A/V profile extension for generic forward error correction in RTP. Work in Progress.

[18] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A transport protocol for real-time applications", RFC 1889, January 1996.

[19] L. Vicisano, L. Rizzo, and Crowcroft J. TCP-like congestion control for layered multicast data transfer. In Proceedings IEEE INFOCOM'98, 1998.

[20] R. X. Xu, A. C. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous media applications. In Proceedingsof the 7th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'97), Washington University in St. Louis, Missouri, May 1997.

[21] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the Mbone multicast network. In Proceedings IEEE Global Internet Conference, November 1996.

Full Copyright Statement