Effects of Interleaving on RTP Header Compression

Colin Perkins Jon Crowcroft
Department of Computer Science
University College London
Gower Street
London WC1E 6BT

Abstract—
We discuss the use of interleaving as a bandwidth efficient means of protecting audio streams from the effects of packet loss in the Internet. The adverse effects of interleaving on IP/UDP/RTP header compression are noted and a number of schemes which remedy this problem are discussed.

Keywords—
RTP, Interleaving, Voice-over-IP, header compression.

I. Introduction

In recent years the market for, and use of, voice-over-IP and other streaming media applications in the Internet has grown at a phenomenal rate. As such services become more widely deployed the limitations of best-effort IP transport are becoming apparent, and it is clearly necessary to provide these media streams with some protection from the worst effects of packet loss.

The Internet Engineering Task Force has responded to this with the definition of a number of payload formats which extend the real-time transport protocol, RTP [1], to provide some degree of error resilience [2–4]. These payload formats share the common feature that they add error correction information to a media stream, gaining protection from packet loss at the expense of increased bandwidth utilization.

Unfortunately, this increase in bandwidth may be unacceptable in many cases, for example low-speed dialup or wireless links may not be able to support the bandwidth requirements of such a stream. Interleaving provides an effective means by which audio streams may be protected which trades latency, rather than bandwidth, for such protection.

There are, however, a number of interactions between interleaved media streams and RTP header compression [5] which reduce the effectiveness of this approach. This paper describes these interactions, and notes possible solutions.

It should be noted that the application of error resilience techniques to best effort IP transport of RTP flows is not the only means by which those flows can be protected. For example, the IETF has also defined the integrated [6] and differentiated [7] services frameworks which can provide various levels of guaranteed quality of service for RTP flows. However, due to the large nature of the change entailed by these new forwarding models, and the difficulties in producing a charging model for such services, deployment of enhanced transport services in the Internet has been slow. For this reason, we do not further discuss the use of such transport in this work, although we do expect it to become important in the future.

The remainder of this paper is structured as follows: in section II we briefly review the use of interleaving, packetization options for interleaved streams and IP/UDP/RTP header compression. The problematic interaction between interleaved media and header compression is noted in section III, with a number of solutions to this being discussed in section IV. Section V and VI discuss related and future work, respectively. Finally section VII concludes the paper.

II. Background

A. The Interleaving Process

An interleaver is a device which permutes the order of a sequence of symbols. The corresponding device which restores the original order of the symbols is a deinterleaver. An interleaver is employed in a transmission system when it is desired to randomize the distribution of errors after recepetion: a burst of loss on the channel is transformed into a sequence of isolated losses by the interleaving process.

A block interleaver is an interleaver where the permutation function repeats periodically. Consider an $n \times m$ matrix representing the mn successive symbols that are stored in a buffer prior to transmission, illustrated in figure 1. Symbols are read into the matrix by rows, starting from the position labelled ‘I’, and read out by columns starting from the ‘O’ position.

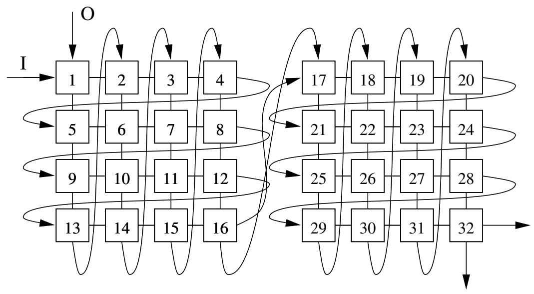

Fig. 1. A sample 4x4 block interleaver

For continuous interleaving two matrices are needed, with symbols being written into one matrix whilst they are read out of the other. This clearly leads to considerable delay in the interleaver, with output of symbols from the buffer matrix being delayed until all symbols have been read in.

The rearrangement of the symbols by the interleaver is such that if m or fewer symbols are lost from a block, each original group of n symbols after deinterleaving will contain at most one loss.

The process by which an interleaver can be applied to a packet audio system is illustrated in figure 2. Codec frames are treated as the symbols on which the interleaver operates and resequenced before packetisation.

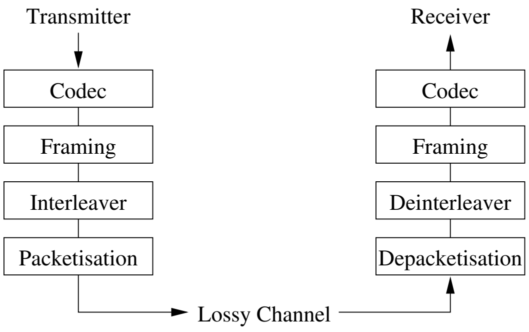

Fig. 2. The transmission process

As an example of the process, consider the 4x4 interleaver illustrated in figure 1 which resequences the codec frames as shown in figure 3. It is important to note that, although frames are sent in a different order to their original creation, this order is not random and has a definite pattern to it (the importance of this will become clear later when we discuss RTP header compression).

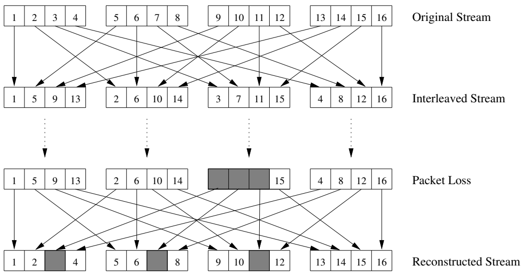

Fig. 3. The interleaving process

It can be seen that a burst of consecutive loss in an interleaved stream will result in multiple small gaps in the reconstructed stream. This spreading of the loss is important for two similar reasons: firstly, packet voice applications typically transmit packets which are similar in length to phonemes in human speech. Loss of a single packet will therefore have a large effect on the intelligibility of speech. If the loss is spread out so that small parts of several phonemes are lost, it becomes easier for listeners to mentally patch-over this loss [8] resulting in improved perceived quality for a given loss rate. In a somewhat similar manner, error concealment techniques perform significantly better with small gaps, since the amount of change in the signal's characteristics is likely to be smaller.

The majority of speech and audio coding schemes can have their output interleaved. Provided the channel exhibits bursts of loss, rather than isolated loss events, interleaving provides an effective means by which a media stream may be protected. The disadvantage of interleaving is that it increases latency. The major advantage of interleaving compared to other means of protecting media streams is that it does not increase the bandwidth requirements of a stream.

B. RTP Packetisation

The protocol of choice for IP-based packet audio applications is the Real-time Transport Protocol, RTP [1]. The RTP header has the following format:

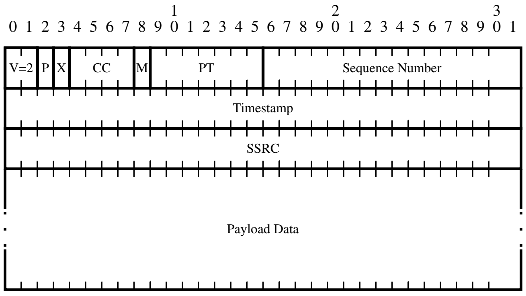

Fig. 4. RTP Packet Header

The important features of the RTP header, when discussing interleaving, are the sequence number and timestamp. An RTP packet may contains one or more codec frames, with the sequence number incrementing by one for each packet sent and the timestamp increasing at the rate of the sampling clock. For example, if using a codec with 20ms frames at 8kHz sampling, for example GSM, the timestamp will increase by 160 (8000 samples per second $\times$ 0.020 seconds) with each packet sent, assuming one frame per packet.

An RTP receiver uses the sequence number to detect lost packets and the timestamp field to determine when to playout received data. This allows for interleaving without changing the basic decoder model and without the receivers being aware that interleaving is in use, provided only a single frame is included in each packet.

When transmitting an interleaved stream, the interleaving process takes place on codec frames, before the RTP headers are generated. As before, the sequence number is incremented by one for each packet sent, but the timestamps in the RTP headers follow the interleaved codec frame order. If each RTP packet contains a single codec frame, it is a simple matter for the receiver to reconstruct an interleaved stream; frames are decoded in the order specified by the RTP timestamp. In addition, the interleaving function and codec can change at any time, without requiring additional signalling.

If multiple frames are packed into each RTP packet, the timestamp is not sufficient for the receiver to reconstruct the media stream. It is also necessary to convey the order in which frames are packetized. This is discussed further in section IV-C.

For now we will consider the case where a single codec frame is packed into each RTP packet, since this is expected to be the most common case.

C. RTP Header Compression

RTP header compression [5] relies on the exploitation of predictable differences between consecutive packet headers to compress the 40 octet UDP/IP/RTP header down to 2 octets, in the best case.

In TCP/IP header compression, the gain comes from noting that many header fields are identical between consecutive packets, and may be elided after the first packet. Differential coding reduces the size of the remaining fields. This concept is extended in the RTP/UDP/IP compression algorithm, where it is noted that, whilst many fields change from packet to packet, the

Packet	Frame	RTP Timestamp	Increment	
1	1	0		
2	5	640	640	
3	9	1280	640	
4	13	1920	640	
5	2	160	-1760	*
6	6	800	640	
7	10	1440	640	
8	14	2080	640	
9	3	320	-1760	*
10	7	960	640	
11	11	1600	640	
12	15	2240	640	
13	4	480	-1760	*
14	8	1120	640	
15	12	1760	640	
16	16	2400	640	
17	17	2560	160	*

Fig. 5. Values of header fields for the example interleaved stream illustrated in figure 1. Note that in practice the timestamp starts from a random value, rather than from zero as illustrated. This does not affect the result.

change is constant and therefore the second order difference is identical. By noting this, and passing the first order difference with the first packet, these fields may also be elided in subsequent packets.

III. Problem Statement

The standard packetization of an interleaved RTP packet stream results in a sequence of packets where the second order difference between the timestamp fields in the header is non-zero in some cases. This has adverse affects on RTP header compression.

As an example of this, consider the interleaving scheme used in our previous examples (figure 1). The mapping from packet order to frame order and the corresponding RTP timestamp is shown in figure 5. Those packets denoted by * have non-zero second differences between consecutive values of the timestamp field, and hence do not compress well.

It should be noted that whilst the precise timestamp increments - and hence the overhead due to poor compression - depend on the interleaver in use, the problem exists in some form with all interleavers.

With the rise of personnal cellular telephony systems, and increased interest in the convergence of voice-over-IP systems with the traditional telephone network, it becomes clear that it is desirable to use RTP header compression with interleaved packet streams [9]. A cellular radio transmission network has burst loss characteristics to which interleaving is an effective counter, and the low bandwidth available calls for header compression.

For reasons such as these, it is desirable to develop a means by which RTP header compression can be made to function well with interleaved streams.

IV. Possible Solutions

There are three possibilities for dealing with the problem of non-uniform timestamp increments:

1. Do nothing, based on the assumption that header compression works well enough despite this.
2. Modify the header compression scheme to recognize interleaved streams, and to expect this variation in timestamp.
3. Packetize the stream differently, to avoid this variation in timestamp.

Clearly options 1 and 3 are simplest to implement, since they require either no change, or a change to end-point RTP implementations only. Option 2 is harder to deploy, since it also requires changes to router code.

We now discuss these three possibilities in more detail.

A. *How well does header compression work?*

The factor which limits the amount of header compression achievable for an interleaved RTP audio stream is the non-constant variation in timestamp between packets. There are three factors which influence this:

- the media clock, s samples per second
- the packetisation interval, i seconds per packet
- the interleaving function, an $n \times m$ matrix

Since block interleavers have periodic output, we may calculate the repeating pattern of timestamp increments for an interleaved packet stream according to these parameters, and hence characterize the efficiency of the header compression algorithm for a particular interleaver over its repeat cycle.

If interleaving is not used, the timestamp increment between packets is $t = si$ samples per packet. Since interleaving does not change the size of packets, the increments for an interleaved stream must be multiples of this value.

Recall that, for a $n \times m$ block interleaver, symbols are read in as n rows of m symbols and read out as m columns of n symbols. This results in a periodic pattern to the timestamp increments: within each column the increment is constant, but differs for the first packet from each column. In addition, the first packet in each block has a different increment from all the others in that block. The process is illustrated in figure 6. The timestamp increment for the first packet of each block is t samples. The first packet read from each column of the interleaving matrix therefore has a timestamp increment of $-((n-1)m-1)tm$ samples, and each of the $n-1$ other packets within the column has a constant increment of mt samples.

Those packets with the same timestamp increment as the previous packet may be sent with fully compressed two octet headers. Those where the timestamp increment differs from the previous packet are sent with a compressed RTP header including the timestamp increment (3-5 octets, depending on the value of the increment [5]).

For an $n \times m$ block interleaver we observe that there are $2m$ packets which must be sent with an increment, comprising

- 1 packet where the timestamp increment is t samples (at the crossover point between two interleaving matrices).
- $m-1$ packets for which the timestamp increment is $-((n-1)m-1)t$ samples (the first packet in each column).
- m packets for which the timestamp increment is tm samples (the second packet in each column, to set the increment for the remainder of the column).

The remaining $(n-2)m$ packets from each round of the interleaver may be fully compressed.

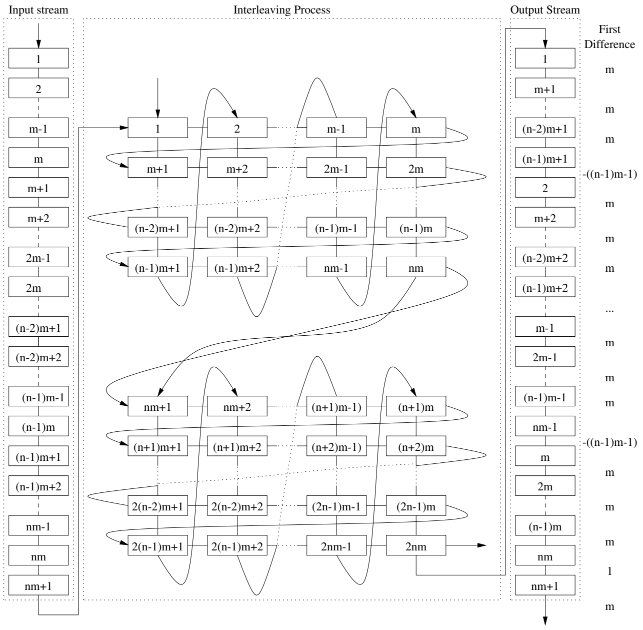

Fig. 6. Timestamp increments due to interleaving

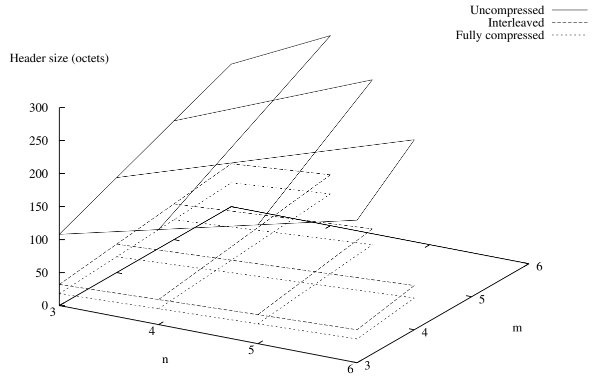

Fig. 7. Total header size for an interleaved stream, compared to a fully compressed stream (2 octet headers) and a non-compressed stream (12 octet headers), for a single repeat cycle of various $n \times m$ block interleaver configurations. Missing entries require timestamp increments outside the representable range of the standard delta encoding table.

If we assume a particular sampling rate and packetisation interval, for example 8kHz with 20ms packets, the factor t becomes constant and we may plot the overhead caused by the interleaving, measured by packet header size, as is shown in figure 7. It is clear that, whilst the overheads are significantly reduced compared to the uncompressed case, interleaving has a pronounced effect on the performance of the header compression algorithm, particularly when large interleaving matrices are used.

The use of other sampling rates and packetisation intervals will change the constant offset, but not the shape of the curves.

B. Can we modify header compression?

Since we have determined that the standard header compression algorithm performs poorly in the presence of interleaved media, it becomes necessary to seek an improved compression algorithm. That is, to modify the header compression scheme to recognise interleaved streams and to expect, and code for, the inherent timestamp variation.

The obvious means by which the header compression can be improved is to modify the default delta encoding table such that the timestamp differences produced by the interleaving process can be efficiently encoded in a single octet each, without using any multi-octet combinations. This leads to the set of header sizes illustrated in figure 8. It is clear that such a custom delta encoding table leads to a significant improvement in the performance of the header compression scheme, relative to the standard delta encoding table. It also allows offsets outside the range of the standard table to be represented.

	$m=3$	$m=4$	$m=5$	$m=6$
$n=3$	24/32	32/43	40/54	48/65
$n=4$	30/38	40/51	50/64	60/77
$n=5$	36/44	48/59	60/74	72/—
$n=6$	42/50	56/67	70/—	84/—

Fig. 8. Header size for the header compression scheme with custom delta encoding table, compared to the standard delta encoding table, for a single repeat cycle of various block interleaver configurations. Entries in the columns comprise the sizes of the headers: custom table/standard table.

It should be noted that the concept of loading a non-default delta encoding table into the header compression engine is explicitly described in [5] where "...it is recommended that implementations use a table-driven delta encoder and decoder to allow negotiation of a table specific for each session if appropriate, possibly even an optimal Huffman encoding". However, that specification does not indicate how such a non-default encoding table may be loaded, leaving that to the link layer protocol.

As noted previously, the timestamp deltas which are required to efficiently compress a block interleaved packet stream may be derived from the interleaver parameters. Therefore, if a means of loading the new compression table may be found, such gains may be readily achieved. Possible means by which the encoding table may be loaded are discussed later.

A more impressive gain can be achieved by modifying the header compression scheme to be explicitly aware of the interleaving process. In this case, details of the interleaving matrix are communicated out-of-band to the hosts performing the header compression. There is then no need to transmit the timestamp delta since those hosts explicitly keep track of the position in the interleaving sequence and generate RTP timestamps in the decompressed packets to match this. Unlike the change to the delta encoding table discussed above, this is a significant change to the header compression model, but it has the potential to achieve higher compression gain.

The header compression engines support a context identifier to differentiate multiple packet flows and a 4 bit per flow sequence number used to detect packet loss between the compressor and decompressor. Packet loss on the link for which compression is being performed is assumed to be rare, but a mechanism is provided by which the receiver may indicate that loss

```
int dt[] = {-1760, 160, 640, 0};
int fd;
...
setsockopt(fd, SOL_SOCKET, SO_RTPCOMPRESSTABLE,
           (void *)dt, sizeof(dt));
```

Fig. 9. Use of setsockopt() to set the delta encoding table

occurred, and request that the sender transmits a packet with an uncompressed header to refresh the state.

We can leverage this sequence number to determine the current position within the interleaving sequence, provided that the sender and receiver are initially synchronised, that the interleaving sequence is known and that the sequence proceeds without pause or interruption. Once the position within the interleaving sequence is known, the header compression engine can simply send fully compressed packets even when the timestamp increment is non-uniform, provided that the non-uniformity is expected. As noted in section IV-A the performance of the header compression scheme when all headers can be fully compressed is significantly better than that when timestamp deltas have to be communicated.

Both modifications to the header compression scheme require that some information is communicated to those hosts performing the compression. In the first case this is the table of delta encodings, in the second it is the details of the interleaving matrix. There are two possibilities for this communication, depending on the location of the link which requires header compression. If it is the access link from the sender of the interleaved data one can envisage that the application can easily set the appropriate parameters for the compression layer. If the link requiring compression is remote from the sender, the routers at either end of that link must derive the correct encoding table/interleaver parameters themselves.

In the local case we envisage, for example, an extension to the setsockopt() call on systems implementing the Berekely sockets API [10] whichs sets the header compression parameters. This would take as the optval parameter a pointer to the delta encoding table to be used with, perhaps, the syntax shown in figure 9.

In the remote case there are, once again, two options: the routers can either invoke some heuristic to determine that a flow is interleaved, or the end-systems can pass the required information to the routers in some manner.

We do not expect it to be feasible for routers to use heuristics to determine the presence of interleaved flows and to intuit the optimum parameters, due to the amount of state this would require to be kept and the complexity of producing an optimum table and communicating it to the peer router. In particular, there has been some concern expressed that current routers are unable to perform header compression at high rates, and additional complexity here is unlikely to be welcome.

As an alternative to this, a signalling system such as, for example, RSVP [11] may be employed by which end systems can signal to the routers in advance that certain parameters are required for a given flow. Whilst the current RSVP flow-specifications do not allow description of interleaving parameters, we would expect that such parameters can easily be added if desired. This leads to the overhead of implementing RSVP, with its associated state, in routers but should not impact complexity of the fast-path forwarding engine, making this is potentially more desirable solution. It is still a heavy-weight solution, however.

To conclude, we note that there are a number of means by which the RTP header compression algorithm can be modified to better suit interleaved flows. Such schemes are difficult to implement in those cases where compression is performed in the core of the network, but relatively simple extensions to the current model would allow implementation in those cases where end-systems perform compression. Since, in many cases, the low-bandwidth link which would benefit from header compression is at the edge of the network, we believe there is considerable potential in these approaches.

C. Can we packetise the stream differently?

A common technique used to reduce the header overhead in RTP flows is to pack multiple codec frames into a single packet. For example, an application which packs two consecutive GSM frames into a single packet, instead of sending two separate packets, will reduce the data rate from 29.2kbps to 21.2kbps (including headers) due to the absence of the second IP/UDP/RTP header. Such a saving is clearly worthwhile, and it may be possible to achieve similar gains for interleaved flows.

In particular, for an $n \times m$ block interleaver, it is possible to group each of the n frames from each column of the interleaver matrix into a single packet (subject to network MTU constraints, of course). This has two advantages: the number of packets is reduced by a factor of n compared to sending each frames as a single packet, and those packets which are sent have more uniform timestamp increments allowing effective header compression.

When grouping multiple interleaved frames into a single RTP packet it is necessary to convey the playout time for each frame in some manner, since it is not possible to uniquely derive this from the timestamp in the RTP header alone. This may either be done implicitly, using the RTP sequence number to indicate the position in the interleaving sequence, or by explicit inclusion of a timestamp for each frame.

Implicit conveyance of the timestamp is more bandwidth efficient, but requires out-of-band signalling to convey details of the interleaving parameters in use to the receiver, such that it can construct the position in the sequence gven a timestamp and sequence number only. The need for out-of-band signalling can be averted by including an explicit timestamp on each frame as it is packed into a packet, for example it has been proposed [12] to re-use the RTP payload format for redundant audio [2] for this purpose.

If implicit timestamps are used, this alternative packetisation has clear bandwidth savings compared to a stream with one frame per packet. The header overhead on $n-1$ of the n frames from each column of the interleaver matrix is reduced to zero octets (since there is no need to send the headers), and the remaining headers may be compressed using the techniques discussed in section IV-B to two octets each.

If explicit 16 bit timestamps are used, as signed offsets from the timestamp in the RTP header, the result is a stream with the same header overhead as with optimum compression and one frame per packet. The packet rate is reduced by a factor of n, however.

We also note that interleaved streams packetised in this manner exhibit a significant bandwidth saving in the backbone of the network, where header compression is not typically performed.

A significant disadvantage of these alternative packetisations is that the loss of a single packet affects multiple frames (the interleaving process helps, since those frames will not be adjacent in the reconstructed stream). It is unclear whether this loss multiplier effect is significant, or if the n-fold reduction in the number of packets flowing will result in a lower packet loss rate to counteract the effect. This will almost certainly be network dependent.

To conclude, we note that an alternative packetisation of an interleaved media stream can lead to more efficient transport than that achieved by RTP header compression and a more traditional packetisation. In addition, the bandwidth requirements in the core of the network, those links where header compression cannot typically be employed due to processing time limitations in routers, are significantly reduced. On the down side, it is unclear how packet loss affects such a stream and if the reduction in packet rate will lead to a sufficient reduction in packet loss rate to counteract the loss multiplier effect caused by grouping multiple frames into a single packet.

V. Related work

Interleaving by the transmitter is not the only source of reordering in the Internet. It has long been recognised that the network itself can introduce some reordering into packet flows, due to route changes, and this will also have adverse effects on the performance of RTP header compression.

For example, in the large scale survey of Internet packet dynamics by Paxson [13], it is noted that "Internet paths are sometimes subject to a high incidence of reordering, but the effect is strongly site dependent, and correlated with route fluttering". That work notes that packet reordering "rarely had a significant effect on TCP flows, because generally the scale of the reordering was just a few packets" which would imply that there would be relatively few cases in which it would impact RTP header compression.

The reception of interleaved and/or reordered RTP flows must also affect the playout adaptation/estimation algorithm used in the receiver. Playout adaptation for RTP flows is a reasonably well understood area, but most existing work [14, 15] assumes that packets arrive in order, or that reordering is rare. In addition, most existing voice-over-IP applications are designed with the goal of minimizing end-to-end delay, for example by keeping the adaptive playout buffer as small as possible, which conflicts with the requirements for correct playout of interleaved media.

When designing a playout adaptation for an interleaved media stream it is necessary to ensure that it includes sufficient delay to buffer an entire block of the interleaver's output before starting playout. If using the packetisation scheme with explicit timestamps as discussed in section IV-C, it becomes a simple matter to include the required playout delay in each packet enabling the receiver to adapt.

If using a modified header compression scheme such as those in section IV-B or if using a packetisation with implicit timestamps, as in section IV-C, it is necessary to either design a playout adaptation algorithm which can recognize that interleaving is in use or to pass the required playout delay as an out-of-band parameter. This seems to be an area where further work is needed.

VI. Future work

There is an additional class of interleaver which may be employed as an alternative to the block interleavers we have discussed. These convolutional interleavers (see for example [16]) are the subject of further study, but are not expected to significantly alter our results.

As we noted in section IV-C, the loss multiplier effect caused by packing multiple interleaved frames into a single packet is difficult to quantify. There are two areas which need investigation:

- listening tests may be performed to determine the effects of packet loss by comparing subjective quality ratings for streams packetised with both single and multiple frames per packet at varying packet loss rates
- measurements of packet loss rates may be taken over different network connections for both types of stream, to attempt to measure the effects of an n-fold reduction in packet rate on the observed packet loss

It is well known that these two areas are hard problems in their own right, and that meaningful results are difficult to produce. We therefore consider such work to be outside the scope of this present paper, although we intend to pursue the first option in the future. The second option - evaluation in real networks - would not add significantly to the results presented here. Whilst such an evaluation would be a useful indication of the sensitivity of header compression to packet loss, it is orthogonal to the present discussion of header overheads.

VII. Conclusions

The major contribution of this paper is the recognition that RTP header compression and interleaving have undesirable interactions. In addition, we have outlined a number of possible remedies and shown how to calculate the overheads caused due to these interactions.

As a result, we have shown that the IP/UDP/RTP header compression scheme can perform well with interleaved streams, provided that either

- the default delta encoding table is replaced with one tailored to the particular interleaving scheme in use; or
- the compression engine is made explicitly aware of the interleaving process, and can take this into account when predicting values for the timestamp.

Both alternatives require some means of signalling to the compression engine parameters relating to the interleaving function. We have noted that, since header compression is typically performed at the edges of the network, this may readily be achieved using, for example, a new socket option.

If neither of these two modifications can be made to the header compression algorithm, we have shown that an alternative packetisation of an interleaved stream can achieve similar bandwidth efficiency. In addition, such a packetisation is important for efficient bandwidth usage in the core of the network. This alternative may have poor performance in the presence of packet loss, although this is not yet proved.

If one is concerned only with the efficient utilisation of a low-bandwidth link at the edge of the network, our recommended solution is to use one of the modified header compression algorithms, as discussed in section IV-B.

However, it is worth noting that the alternative packetisation discussed in section IV-C has lower overhead when RTP headers are uncompressed, as is the case in the core of a network. Given the recent interest in RTP multiplexing solutions, to reduce the header overhead when trunking many simultaneous calls between IP telephony gateways [17, 18], it is possible that this saving will take precedence over a bandwidth saving at the edges of the network.

ACKNOWLEDGMENTS

This work was supported by the European Commission Telematics for Research project RE4007.

REFERENCES

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," January 1996, RFC1889.

[2] C. S. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J.-C. Bolot, A. Vega-Garcia, and S. Fosse-Parisis, "RTP Payload for redundant audio data," November 1997, RFC2198.

[3] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction," IETF Audio/Video Transport Working Group, November 1998, Work in progress.

[4] J. Rosenberg and H. Schulzrinne, "An RTP payload format for Reed-Solomon codes," IETF Audio/Video Transport Working Group, November 1998, Work in progress.

[5] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP headers for low-speed serial links," February 1999, RFC2508.

[6] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," June 1994, RFC1633.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," December 1998, RFC2475.

[8] G. A. Miller and J. C. R. Licklider, "The intelligibility of interrupted speech," *Journal of the Acoustical Society of America*, vol. 22, no. 2, pp. 167–173, 1950.

[9] Mikael Degermark (Editor), "3GPP requirements for header compression," October 1999, Exerpt from 3GPP TR 23.922 version 1.0.0, 3rd Generation partnership Project; Technical Specification Group Services and Systems Aspects; Architecture for an All IP network. Input to the I-ETF robust header compression BOF, November 1999.

[10] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," September 1997, RFC2205.

[12] C. S. Perkins, "RTP payload format for interleaved media," IETF Audio/Video Transport Working Group, February 1999, Work in progress.

[13] V. Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 1–16, August 1999.

[14] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings IEEE INFOCOM*, Toronto, Canada, June 1994.

[15] I. Kouvelas and V. Hardman, "Overcoming workstation scheduling problems in a real-time audio tool," in *Proceedings of the USENIX Annual Technical Conference*, Anahiem, CA, January 1997.

[16] J. L. Ramsey, "Realization of optimum interleavers," *IEEE Transactions on Information Theory*, vol. IT-16, pp. 338–345, May 1970.

[17] B. Subbiah and S. Sengodan, "User multiplexing in RTP payload between IP telephony gateways," IETF Audio/Video Transport Working Group, August 1998, Work in progress.

[18] K. Tanigawa and T. Hoshi, "Simple RTP multiplexing transfer methods for VoIP," IETF Audio/Video Transport Working Group, November 1998, Work in progress.