Realisation of an Adaptive Audio Tool

Arnaud Meylan and Catherine Boutremans

Institute for Computer Communication and Applications
Swiss Federal Institute of Technology at Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Abstract. *Real-time audio over the best effort Internet often suffers from packet loss. So far, Forward Error Correction (FEC) seems to be an efficient way to attenuate the impact of loss. Nevertheless to ensure efficiency of FEC, the source rate must be continuously controlled to avoid congestion. In this paper, we describe a realisation of adaptive FEC subdued to a TCP-friendly rate control.*

1 Introduction

The Internet has changed from mainly a file transfer and e-mail tool to a network for multimedia and commercial applications, among others. This change brought up many new technical challenges such as transport of real-time data over non real-time lossy networks, which has been fulfilled by the Real-time Transport Protocol (RTP) [8] and FEC techniques [3].

Unfortunately, FEC is too often used without rate control, what leads to more congestion, loss and then worse audio quality [6]. The purpose of this work is to add adaptive FEC to an existing software: the Robust Audio Tool [7]. FEC will be constrained by a TCP-friendly rate, proposed by Mahdavi and Floyd [16].

In this paper the general problem of optimizing quality at reception is presented, a more specific solution is deduced.

2 State of the Art

2.1 FEC

Definition Forward Error Correction (FEC) relies on the addition of repair data to a stream, from which the content of lost packets may be recovered at destination, at least in part. Two classes of repair data may be added to a stream [3]: those which are independent of the contents of that stream (e.g. Parity coding, Reed-Solomon codes) and those which use knowledge of the stream to improve the repair process. In the context of real-time audio the most popular scheme, which was standardized by IETF is repetition of audio units on the stream. In the following, we will refer to this scheme as Signal processing based FEC (SFEC).

SFEC In order to simplify the following discussion, we distinguish a (media) *unit* of data from a *packet*. A unit is an interval of audio data, as stored internally in an audio tool. A packet comprises one or more units, encapsulated for transmission over the network.

The principle of FEC used here is to repeat units of audio in multiple packets. If a packet is lost then another packet containing the same unit will be able to cover loss and be played—providing it arrives. This approach has been advocated in [4] and [5] for use on the Mbone, and extensively simulated by Podolsky [6] who calls this framework "Signal-processing based FEC" (SFEC[1]). Redundant audio units are piggy-backed onto a later packet, which is preferable to the transmission of additional packets, as this decrease the amount of packet overhead and routing decisions.

There is a potential problem with apply redundancy in response to loss because when loss occurs the (user) reflex is to add redundancy at the source. This leads to increase the overall rate transmission and congestion, probably producing even worse quality. Effectively, redundancy can be added when loss occurs, but in the same time the source encoding must be changed to use less bandwidth in response to congestion [6]. Our framework is to continuously ensure that the overall rate transmission is smaller or equal than a TCP-friendly rate proposed by Mahdavi and Floyd [16]. It is a combined source rate/redundancy control.

2.2 TCP-Friendly Rate Control

As networked multimedia applications become widespread, it becomes increasingly important to ensure that they can share resources fairly with each other and with current TCP-based applications, the dominant source of Internet traffic. The TCP protocol is designed to reduce its sending rate when congestion is detected. Networked multimedia applications should exhibit similar behavior, if they wish to co-exist with TCP-based applications.

One way to ensure such co-existence is to implement some form of congestion control that adapts the transmission rate in a way that *fairly* shares bandwidth with TCP applications. One definition of *fair* is that of TCP *friendliness* [16] - the non-TCP connection should receive the same share of bandwidth (namely achieve the same throughput) as a TCP connection.

Mahdavi and Floyd [16] have derived an expression relating the average TCP throughput R_{TCP} to the packet loss rate:

$$R_{\mathrm{TCP}} = 1.22 \frac{\mathrm{MTU}}{\mathrm{RTT} \times \sqrt{\pi_1^*}} \quad . \tag{1}$$

where MTU is the packet size being used on the connection; RTT is the round trip time and π_1^* is the loss rate being experienced by the connection.

In our application, we fix the MTU to 576 bytes, the minimum size for TCP. Current values for RTT are obtained using RTCP Sender and Receiver Reports.

[1] "because this approach exploits a signal-processing based model of the audio signal to effectively compute its error-correcting information"

The packet loss rate π_1^* is computed at the receiver and reported to the sender via the Fraction lost field of the Receiver Reports.

3 The Robust Audio Tool

3.1 Software Presentation

The Robust Audio Tool (RAT) [7] is an open-source audio conferencing and streaming application that allows users to particpate in audio conferences over the Internet.

This software is based on IETF standards, it uses RTP above UDP/IP as its transport protocol, according to the RTP profile for audio and video conference with minimal control [8][9]. RAT features a range of different rate and quality codecs, receiver based loss concealment to mask packet losses, and sender based channel coding.

3.2 Some Useful Concepts about RAT

The channel coder Let us define the following terms:

A *codec* is defined as an encoder-decoder performing one type of compression/decompression on the audio stream. The encoder part takes a buffer of audio stream in input, performs encoding and outputs a playout buffer of *media units*.

The *channel coder* builds *channel units* which represent the payload of the RTP packets. In RAT the channel coder performs four different kind of channel coding, especially three allowing FEC called *redundancy* (SFEC), *interleaving* and *layered*. For our work, FEC will be added in the form of SFEC, using the redundant channel coder.

The Message Bus RAT comprises three separate processes: controller, media engine and user interface. Communication between them is provided by the Message bus (Mbus), the sole means to ensure coordination of multimedia conferencing systems.

The Message bus was proposed by Colin Perkins and Jorg Ott [10]. It solves the typical problem of separate tools providing audio video and shared workspace functionality. It maps well on the underlying RTP media streams, which are also transmitted separately

A message contains a header and a body. The former indicates notably the source and destination adresses the latter contains the message having to be delivered to the application. The message is transmitted by UDP in the form of a string. A function maps it into a C function call at destination (unmarshalling).

We considered the following MBus commands:

Command name	*Usage*
`tool.rat.codec(.)`	Specifies primary codec being used by this source
`audio.channel.coding(.)`	Specifies secondary codec, and its relative offset to the primary
`tool.rat.rate(.)`	Sets the number of audio units (codec frames, typically) placed in each packet when transmitting, assuming a unit is t ms long

These messages provide high level access to the program. The biggest part of implementation was made using these commands.

4 Adaptive FEC Algorithm

4.1 The General Problem

We want to maximize a certain measure of quality at reception. Quality can be estimated according to a non-subjective (loss rate after reconstruction) or subjective measure (perceived quality at destination depending on the quality of each audio unit played). To realize this, the SFEC-channel coder algorithm must choose a range of variables:

- k denotes the number of copies sent for the same audio unit. k is bound to the loss probability of the network and desired apparent loss rate after reconstruction at reception.
- $\mathbf{o} = [o_1, o_2, \ldots, o_k]$ denotes the offset of redundant unit i relative to the first transmitted unit. One always have $o_1 = 0$.
  It has been shown [11] that loss presents some degree of correlation: if packet n is lost, then there is a higher than average probability that packet $n+1$ will also be lost. This indicates that an advantage in perceived audio quality can be achieved by offsetting the redundancy. The disadvantage of offsetting is the increase of the playout delay. It's a compromise between delay and robustness. Nevertheless in case of half duplex for example, delay is not important and offset can be extensively used.
- The set of codecs available is associated with a finite set of encoding rates r_i: $\mathcal{R} = \{r_i\}_{i=1}^{n}$. Notice $r_0 = 0$ corresponds to no coding. Codecs are ordered according to their bit rates, namely $i < j \Leftrightarrow r_i \leq r_j$.
- Let $\mathbf{x} = \{x_i\}_{i=0}^{k-1}$ denote the number of codec used to encode the i^{th} redundant unit.
- r is the total payload rate $r = \sum_{i=1}^{k} r_{x_i}$
- t denotes the frame duration of an audio unit, it is usually 20 ms, 40 ms or 80 ms. It influences end to end delay to increase t and reduces the number of packets sent, so diminishes the header overhead.

Input parameters are:

- The TCP-friendly rate constraint R which is slightly smaller than R_{TCP}. Indeed, R_{TCP} provides an upper bound to the *total* throughput $R_{\text{TCP}} = R+$

R_{headers} allowed for the application. R only represents the maximum payload (audio) throughput. R_{headers} is the IP/UDP/RTP header throughput: this application sends (20 + 8 + 12) bytes of header each t ms . The payload rate is then

$$R = R_{\text{TCP}} - \frac{40 \cdot 8}{t} \text{ [kb/s]}. \qquad (2)$$

– The parameters of the loss process on the network, typically p and q for the Gilbert model.

In this context, it would be useful to maximize a function $f(k, \mathbf{o}, \mathbf{x}, t, R, p, q)$ representing the quality at reception. Our solution will be less general, the (restrictive) hypothesis are presented below.

4.2 Models

Loss Model The characteristics of the loss process of audio packets are important to determine how to use SFEC. Previous work on the subject [11], [13], propose a two state Markov Chain, the Gilbert model for the loss process (Fig. 1). States 1 and 0 respectively correspond to a packet loss and packet reaching destination. Let q denote the probability of going from state 1 to state 0, and p from state 0 to state 1. The stationary distribution of this chain is $\pi_0^* = \frac{q}{p+q}$ $\pi_1^* = \frac{p}{p+q}$.

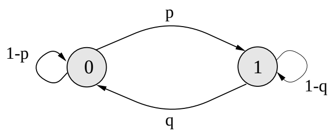


Fig. 1. The Gilbert Model

The actual implementation of the software does not report q. An assumption is then needed. Usually one assumes $p+q = 1$, and the model turns in a Bernoulli of parameter (π_0^*) loss process in which losses are independent. Since this is not realistic [11], we prefer to assume that the average packet loss length is 1.5, therefore $q = 2/3$. This ambitious assumption is based on [14][15], and presents the advantage to reflect the verified correlation of loss process. π_1^* represents the average loss rate, such as reported by RTCP: $\pi_1^* = \frac{p}{p+2/3}$. So p is:

$$p = \frac{2}{3} \cdot \frac{\pi_1^*}{1 - \pi_1^*} \quad . \qquad (3)$$

Loss Rate after Reconstruction We are then interested in loss rates after reconstruction, as a function of k and $\mathbf{o}$. At this stage, a drastic simplification due to actual implementation of RAT appears: k cannot exceed 2, only one level of redundancy can be used. Therefore only o_2 remains to compute.

Let us examine the loss rate after reconstruction for different offsets (we will denote o_2 by n). The probability ρ to lose packet l and $l+n$ is easy to compute using basic probabilities and Markov chains properties

$$\begin{aligned}\rho &= P(\text{pack } l \text{ lost} \cap \text{pack } l+n \text{ lost}) \\ &= (\text{pack } l+n \text{ lost} \mid \text{pack } l \text{ lost})P(\text{pack } l \text{ lost}) \\ &= P(\text{pack } n \text{ lost} \mid \text{pack } 0 \text{ lost})P(\text{pack } 0 \text{ lost}) \\ &= p_{11}^n \pi_1^* \ . \end{aligned} \tag{4}$$

Where p_{ii}^n is the probability to return in state i after n steps, i.e. it is the ii^{th} entry of the n step transition matrix P^n.
For this chain one can compute [12] that

$$P^n = \frac{1}{p+q}\begin{bmatrix} q & p \\ q & p \end{bmatrix} + \frac{(1-p-q)^n}{p+q}\begin{bmatrix} p & -p \\ -q & q \end{bmatrix} \ . \tag{5}$$

hence

$$p_{11}^n = \frac{1}{p+q}(p + q(1-p-q)^n) \ . \tag{6}$$

Therefore the probability to lose both packets is

$$\rho = \frac{p}{(p+q)^2}(p + q(1-p-q)^n) \ . \tag{7}$$

The objective is to minimize ρ since it represents the probability to fail to repair the audio stream. The only adjustable parameter is n (positive integer) since p is bound to the network's state, and we assumed a constant value for q, therefore we compute

$$\frac{\partial\rho}{\partial n} = \frac{pq}{(p+q)^2}(1-p-q)^n \ln(1-p-q) \ . \tag{8}$$

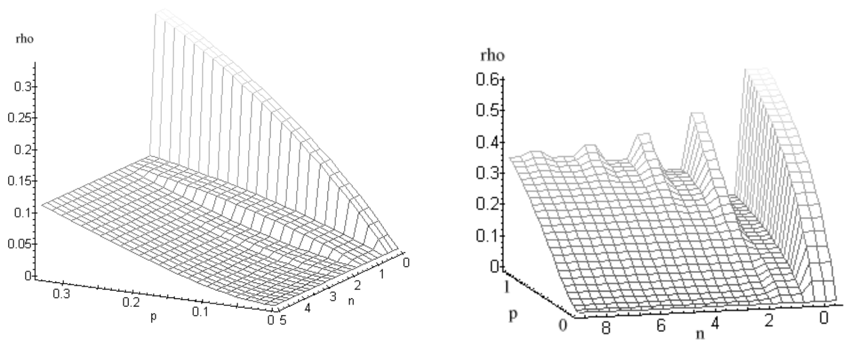
Extrema are obtained for zeros of (8). Three cases are distinguished:

- $\mathbf{0 < 1 - p - q < 1}$ All members of (8) are positive, for $n \to \infty$, $\frac{\partial\rho}{\partial n} \to 0$. $n \to \infty$ leads to a minimum, we do not prove it here. This means that the more you offset for the redundant audio unit the smaller the loss rate after reconstruction. But in the same time playout delay increases since the application must "wait" for the redundant units.
- $\mathbf{1 - p - q = 0}$ All values of n are optimal since ρ is then constant.
- $\mathbf{-1 < 1 - p - q < 0}$ The logarithm function is not defined. We constat on (7) that the smallest value of ρ is obtained for $n = 1$.

Conditions can be simplified in: $p < 1/3$, $p = 1/3$, $p > 1/3$ since $q = 2/3$.

Figure 2 shows plots of ρ as a function of n and p, first one illustrates case $p < 1/3$ second one all three.

On Fig. 2 (a) we can constat that when n increases ρ tends fast to its limit. For $n = 3$, ρ already quite equals the limit. More offsetting probably brings a



(a) (b)

Fig. 2. ρ as a function of n and p. (a) $p \in [0, 0.33]$, (b) $p \in [0, 1]$

small reduction of ρ while it increases playout delay. But we do not precisely know which k is optimal since penalty induced by delay on audio quality is not clearly defined. This explains the need for the above mentioned function appraising quality at reception. Without it, a quite arbitrary choice will be to set $n = 3$. Nevertheless it nicely improves the loss rate after reconstruction compared to $n = 1$ and the playout delay introduced seems reasonable[2].

Two last cases are clear. When $p > 1/3$ one clearly chooses $n = 1$ since it guarantees the smallest delay and loss rate after reconstruction. For $p = 1/3$ any value of n is optimal, so we choose $n = 1$ since it minimises delay. We implemented such an algorithm to select offset of redundand unit:

```
p = 2/3 * π1*/(1 − π1*); //π1* denotes loss rate reported by RTCP
if(p < 1/3) n = 3;
else n = 1;
```

These results are specific to this value of q in the general case, the if should be with $p + q$.

Redundancy Level At this point, we have to decide if redundancy must be used or not, depending on loss rate perceived at reception and the target loss rate at destination τ. In absence of redundancy the loss rate equals π_1^* , if redundancy is used the loss rate after reconstruction is ρ.

It is not very clear how much audio loss can be tolerated at destination, it depends on voice reconstruction techniques, audio unit's length and the subjective tolerance of user. It appears [11] that with packet repetition techniques 5% loss rate after reconstruction can be tolerated. We assume then $\tau = 0.05$ and

[2] extra playout delay due to offset is then 3·frame duration

choice about redundancy becomes:

```
if (π₁* > τ) k = 2;
else k = 1.
```

It is interesting to remark that with this model and one single piece of redundant audio offseted, loss rates on the network up to 22% leads to less than 5% of loss rate after reconstruction. 30% network loss gives 9%. With these theoretical results more than one level of rendundancy seems useless.

Encodings The last parameters to determine are the codecs used for encoding. We use a Free-Phone [13] inspired algorithm which solves the problem assuming a variable number of redundancy levels can be used. Since we have one piece of redundancy at most we derive a simpler algorithm that uses their general result: "The main information[3] should be encoded using the highest quality encoding scheme (among those used to encode the main and the redundant information)".

A set of codecs must be chosen before running the algorithm. We picked one performing encoding in the range of 5.6 kb/s to 64 kb/s with 8 kHz sampling frequency: LPC (5.6 kb/s), GSM (13.2 kb/s), G726-16 (16 kb/s), G726-24 (24 kb/s), G726-32 (32 kb/s), G726-40 (40 kb/s) and A-law (64 kb/s). It represents a quite homogeneous distribution of usable rates, so ensures that available bandwidth will be correctly used.

We ordered the codecs according to their bit rates, this differs slightly from their order of quality. One assumes that choosing a codec with higher bit-rate provides better audio quality.

k denotes the redundancy level ($k \in \{1;2\}$ here) x_i is the codec number for the i^{th} copy, R is the available throughput and r_i is rate used by i^{th} codec assuming N codecs are available. The following algorithm is valid for any $k \geq 1$, it gives the codec number to use for i^{th} redundancy level $i = 1 : k$.

```
x_0 = 1; // ensures minimum encoding
for(i=1:k-1:i++) x_i = 0; r = r_{x_0};
i = 0;
do{
   if(x_i < N) x_i++;
   r=0;
   for(j=0; j<k; j++){
         r += r_{x_j};
         }
   if (r > R){
         if(x_0==1 && x_1==0) break;
         x_i--;
         break;
         }
```

[3] the first audio unit sent

```
if(x_{k-1} = N) break; // all codecs have the best quality
i = ++i % k;
}while(r < R);
```

Frame duration We chose $t = 20$ ms as frame duration. It enables smaller latency and reduces time to wait to receive offseted packets.

5 Evaluation

5.1 Results

We have implemented the above described adaptive channel coding subdued to rate control. Tests were performed between EPFL (Switzerland) and two destinations in Europe: the University College of London and the Faculté Polytechnique de Mons in Belgium.

During one week connections were established with UCL at different times of the day. The loss rates never significantly exceeded 1% so SFEC was not needed and the best possible primary quality was chosen by the algorithm. The absence of loss is certainly explained by the very high speed links available beetween both universities.

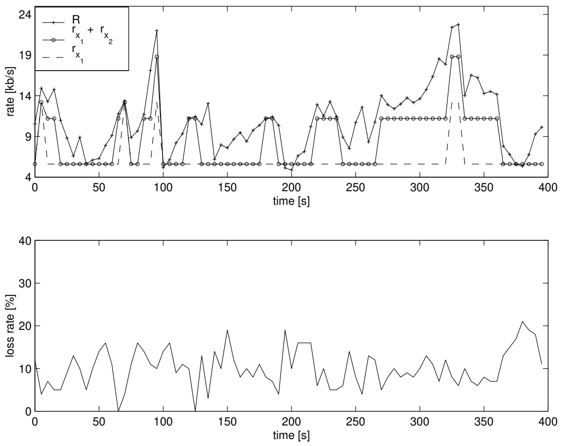


Fig. 3. Source rates and network loss

Connections with the Faculté de Mons were more lossy, the network was in a high degree of congestion since the RTT was about 1.3 sec. and the average loss rate above 10%. Figure 3 shows the TCP-friendly rate constraint R the payload rate $r_{x_1} + r_{x_2}$ and the rate used by the primary encoding r_{x_1} during a 400 sec. connection. This permits to see when redundancy is applied in response to loss on the network which is presented on the bottom.

The rate constraint is respected except when smaller than 5.6 kb/s because our algorithm keeps sending audio at this rate. In this situation the converstion should perhaps be stopped. When redundancy is needed bandwith is shared between primary and secondary encoding. In some cases redundancy should be used but a low rate constraint prevents it.

We could hear a fine quality improvement at destination when using SFEC to attenuate the effect of loss. We do not provide measurements of loss rate after reconstruction here since the efficiency of the method was demonstrated by [13] and [6].

5.2 Future Work

The most important thing to do seems to build a function to estimate quality perceived at reception. Without it, choices must be done statically. With it we could change dynamically offset or frame duration, and probably provide better quality.

Lots of hypothesis where made to build the model, the roughest was certainly to assume $q = 2/3$. It would be useful to have an estimator for this value, computed at destination and reported by RTCP.

In rare cases (loss rates $> 22\%$), one more level of redundancy could be useful. RAT's implementation does not support this at the present time but it seems possible to add this functionality. Notice that then the presented model must be completed to compute the probability to lose all three packets. In our opinion more than three levels of redundancy are not useful.

We stated that a 64 kb/s (A-law) codec provides better quality than a 32 kb/s (G726-32), in our experience the difference is really faint. Instead it could be better to double sampling frequency and keep the G726 codec to get 8 kHz bandpass and 64 kb/s rate. It would have the convicing argument to provide larger bandwidth than Plain Old Telephone System.

6 Conclusion

In this paper we proposed a method to provide adaptive source coding for the Robust Audio Tool. It satisfies fair-rate constraints while improving efficiently quality perceived at reception thanks to adaptation of redundancy level, encoding rates and offset. Nevertheless the proposed algorithms are certainly not optimal because we had to make some suppositions.

7 Acknowledgements

We would like to thank RAT hackers, Orion Hodson and Colin Perkins for their active support during RAT's code exploration. We are also grateful to Alain Ruelle who allowed us to perform the tests with the Faculté Polytechnique de Mons.

References

1. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. *RSVP:A new resourse ReSerVation Protocol.* IEEE Network Mag.,vol.7, no 5, pp. 8-18, sept 1993.
2. W. Almesberger. *Scalable Resource Reservation for the Internet.* Ph.D thesis number 2051, EPFL, November 1999.
3. C. Perkins O. Hodson V. Hardman. *A Survey of Packet-Loss Recovery Techniques for Streaming Audio.* IEEE Network Magazine, September/October 1998.
4. V. Hardman, M. A. Sasse, M. Handkey, and A. Watson. *Reliable audio for use over the Internet.* In Proceedings of INET'95, June 1995, Honolulu, Hawaii.
5. J.-C. Bolot, and A. Vega-García. *The case for FEC-based eror control for packet audio in the Internet.* to appear in ACM/ Springer Multimedia Systems.
6. M. Podolsky, C. Romer, and S. McCanne. *Simulation of FEC-Based error control for packet audio on the Internet.* In Proceedings of IEEE Infocom'98, April 1998.
7. Robust Audio Tool. *http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/.*
8. H.Schulzrinne. *RTP: A Transport Protocol for Real-Time Applications. RFC 1889.* IETF 1996.
9. H.Schulzrinne. *RTP profile for audio and video conferences with minimal control. RFC 1890.* IETF 1996.
10. C. Perkins, and J. Ott. *A message bus for conferencing systems.* http://www.cs.ucl.ac.uk/staff/c.perkins/confbus/.
11. I. Kouvelas, O. Hodson, V. Hardman, and J. Crowcroft. *Redundancy control in Real-Time Internet Audio Conferencing.* In proceedings of International Workshop on Audio-Visual Services over Packet Networks (AVSPN97), September 1997, Aberdeen, Scotland.
12. P. Thiran. *Processus Stochastiques pour les communications.* Module 6, p. 120 EPFL 2000 http://icawww1.epfl.ch/cours_thi/index.html.
13. J.-C. Bolot, S. Fosse-Parisis, and T. Towsley. *Adaptive FEC-Based Error Control for Internet Telephony.* In Proceedings of IEEE Infocom'99, March 1999, New York.
14. A. Vega-García. *Mecanismes de controle pour la transmission de l'audio sur l'internet.* PhD thesis, Université de Nice-Sophia Antipolis, October 1996.
15. D. Ratton Figueiredo and E. de Souza e Silva. *Efficient Mechanisms for Recovering Voice Packets in the Internet.* In Proceedings of IEEE Globecom'99, December 1999, Rio de Janeiro, Brazil.
16. J. Mahdavi and S. Floyd. *TCP-Friendly Unicast Rate-Based Flow Control.* Draft posted on end2end mailing list, Jannuary 1997. http://www.psc.edu/papers/tcp_friendly.html.