Realisation of an Adaptive Audio Tool

Arnaud Meylan and Catherine Boutremans
Institute for Computer Communication and Applications (ICA)
EPFL (Swiss Federal Institute of Technology), CH-1015 Lausanne

March 31, 2000

Abstract

Real-time audio over the best effort Internet often suffers from packet loss. At this time, 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 Forward Error Correction techniques (FEC) [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].

The general problem of optimizing quality at reception is presented, a more specific solution is deduced. A short presentation of software's architecture should help people interested in improving this work.

2 State of the art

This section is a survey of useful techniques for audio exchange over a network.

2.1 Audio coding

Audio applications on the network need two distinct processing: encoding-decoding and packetization-transmission.

Encoding transforms the audio signal which is usually analog to a digital version, with a given rate and quality. Three parameters affect the output rate: sampling frequency, resolution, and the compression means. The audio digital data is put in a packet that is transported through the network. At destination, digital data is decoded—usually the inverse encoding treatment is applied— and played on the audio device. We do not speak of encoder and decoder, but simply of codec (CODer-DECoder), seeing that both actions are bound. Each codec uses a specific method to perform encoding, in our application two principal classes can be outlined:

- The first one uses differential encoding. It is based on the correlation of successive samples of audio signals. Instead of coding the signal, the difference between samples is coded. As they are correlated, the range of the difference is smaller than the range of whole signal.

This enables compression of the signal. Most popular examples for this encoding are DPCM, ADPCM.

- Instead of working with single samples or differences, synthesis coding considers sequences of sound. A model of the source is built[1], and only parameters of the model are transmitted. This enables a drastic compression, but gives quite artificial sound too. Some examples are LPC, and the now famous GSM standard.

We will not give more details on coding techniques in this paper because we borned ourselves to use codecs already implemented in the software. Just remember that the same audio signal can be encoded by different encoders, resulting in different qualities and rates. In our context the rate and quality of the codec are the two important parameters, since the choice of codecs is dictated by the available bandwidth on the network and maximization of audio quality. In the next subsection we present error control schemes.

2.2 FEC

2.2.1 Definition

Audio transmission is highly susceptible to the indeterminacies of the best effort internet service. Packets may be lost, misordered or delayed; this leads to a diminution of the perceived sound quality for the receiver. Clearly, typical error control/loss recovery mechanisms based on the retransmission of lost packets are not acceptable because they dramatically increase end to end latency. Our purpose is to propose a technique able to minimize the impact of loss and delay introduced by transmission.

It seems important to mention that the Quality of Service (QoS) on the Internet is a wide contemporary research topic. Many network researchers think that loss should be explicitly avoided or controlled by re-engineering the network to provide QoS guaranteed, through techniques like admission control and reservation [1], [2]. Nevertheless, our objective is to use the "original" Internet, since it is cheaper and more accessible.

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 second class several means exist. The most popular is to add copies of audio units to the stream, it is also possible to use a special codec that performs layered coding; layers are transmitted in different packets. Only the most popular will be considered here.

2.2.2 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 transmit each unit 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. The principle is illustrated in Figure 1. 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[2]). 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.

[1] codecs designed especially for voice are called vocoders

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

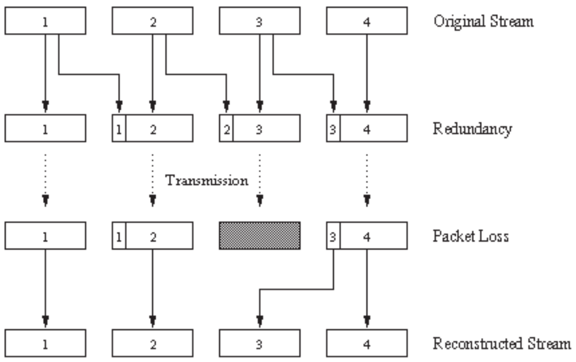

Figure 1: Redundancy principle

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. Two important parameters concerning SFEC are:

- k denotes the number of copies for the same audio unit. In the figure 1, for example, $k = 2$. k is bound to the loss probability of the network and the the 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 unit transmitted. Note that one always have $o_1 = 0$. On Figure 1 $o_2 = 1$.

Algorithms used to adapt redundancy are presented in section 4.

2.3 RTP/RTCP overview

TCP is not appropriated to transmit real-time data since it performs retransmissions and does not offer sufficient support for time management. That is why the IETF developed the *Real Time Transport Protocol* (RTP). It provides end-to-end delivery services for data with real-time characteristics such as interactive audio and video. Those services include payload type identification, sequence numbering, time-stamping and delivery monitoring. RTP also offers a control protocol—RTCP—usually carried on a separate lower level transport association. Below we present a selection of useful fields present in RTP packets:

Payload type Identifies the format of the RTP payload. A profile specifies a default static mapping of payload types codes to payload format. It is also used to identify type of redundancy in the case of audio.

Sequence number The sequence number increments by one for each RTP data sent, and may be used by the receiver to detect packet loss and/or restore packet sequence.

Time-stamp Is derived from a clock at the sender. Reflects the sampling instant of the first byte in the RTP data packet.

Moreover RTCP Sender Reports (SR) and Receiver Reports (RR) provide reception quality feedback (i.e. the fraction of packet lost and the interarrival jitter). They also permit to compute the Round Trip Time (RTT).

2.4 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 congested bandwidth with TCP applications. One definition of *fair* is that of TCP *friendliness* [16] - if a non-TCP connection shares a bottleneck link with TCP connections, traveling over the same network path, 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) to the packet loss rate:

$$R_{TCP} = 1.22 \frac{MTU}{RTT \times \sqrt{\pi_1^*}}$$

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 order to implement a TCP friendly congestion control algorithm, our application will simply choose to send at a rate no higher than the TCP-friendly rate R_{TCP}. The accurate computation of this value requires the application to know the MTU, RTT and current loss rate π_1^*. The MTU may be determined using an MTU discovery algorithm, or assumed to be the minimum acceptable value for TCP of 576 bytes. In our application, we fix the MTU to 576 bytes. Current values for RTT are obtained using the *Time-stamp* fields of the RTCP Sender Reports and Receiver Reports. 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. These can be between two participants directly, or between a group of participants on a multicast group.

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 in the form of redundant audio transmission.

3.2 Some useful concepts about RAT

3.2.1 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. Given such architecture, it is useful to be able to perform some coordination of the separate media tools. For example, it may be desirable to communicate playout-point information between audio and video tools.

A further refinement of this architecture relies on the presence of a number of media engines, and one (or more) user interface agents, which control the media engines and provide a unified conferencing experience. This approach allows flexibility of user interface, but obviously requires some means by which the user interface agent may communicate with the media engines. The Message bus as implemented in RAT provides such a communication channel.

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

Messages are transmitted as UDP messages, an unreliable delivery mechanism. Since it may be necessary to deliver some messages reliably, the message bus allows this through retransmission of lost messages. However most of the messages are sent unreliably, since they are not very important for the application and have a little probability to be lost.

In the next paragraph some useful Mbus commands of RAT are presented. The first ones are quite general , the following are designed specifically for RAT.

Command name	*Use*
`mbus.hello()`	Sent as a heartbeat message every few seconds to indicate liveness
`mbus.quit()`	Sent to indicate the receiving entity should quit
`rtp.cname(.)`	Indicates that the receiver should use the specified cname during this session
More specific commands	
`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(.)`	Set the number of "units" (codec frames, typically) placed in each packet when transmitting, assuming a unit is t ms long
`tool.rat.playout.max(.)`	Set the maximum playout delay allowed
`tool.rat.lecture.mode(.)`	Enables/disables the lecture mode

These messages provide powerful high level access to the program. The biggest part of implementation was made using these commands.[3]

3.2.2 The channel coder

There is a specific vocabulary used for FEC, we will try to explain it. First recall that any audio unit can be encoded with various codecs, and then put in packets. To avoid that kind of vague descriptions, 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*. Typically this buffer is 20, 40 or 60 ms long.

After that a black box performs the preparation of *channel units*, which represent the payload of the RTP packets. In RAT there is a generic framework for this, the *channel coder*. It performs four different kind of channel coding, especially three allowing FEC called *redundancy*, *interleaving* and *layered*. First mode uses scheme described above (SFEC), second one performs a resequencement of audio units before transmission, so that originally adjacent units are separated by a guaranteed distance in the transmitted stream; it disperses the effect of packet losses. The last one uses special codecs, performing a layered coding. There is also a simple mode[4] that simply sends one copy of each media unit. Channel units are then a composing of media units depending on the selected channel coder. For our work, FEC will be added in the form of SFEC, using the redundant channel coder.

[3] For interested programmer, it may be useful to check files `README.mbus_mbus.c, mbus_engine.c, mbus_ui, mbus_control.c`.

[4] Called "vanilla" in RAT's context

4 Adaptive FEC algorithm

4.1 The general problem

In section 2.2 SFEC was presented with it's basic parameters. The objective is to design an algorithm for adaptive channel coding that maximizes a certain measure of quality at reception according to rate constraints. We will have to make some assumptions since the complete problem is non-trivial. The next paragraph explains the most general case.

The general problem is 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:

- As seen below k denotes the number of copies sent for the same audio unit
- $\mathbf{o} = [o_1, o_2, \ldots, o_k]$ denotes the offset of redundant unit i relative to the first transmitted unit is a packet number. 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 $r_0 = 0$, it 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 response to increase t and reduces the number of packets sent, so diminishes the header overhead. It also influences efficiency of receiver audio reconstruction techniques.

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 (headers + payload) allowed for the application. On the other hand, R only represents the maximum *payload (audio) throughput*. We can deduce R from R_{TCP} as follows: $R = R_{TCP} - R_{headers}$, where $R_{hearders}$ is the (IP/UDP/RTP) header throughput which is $R_{headers} = \texttt{header\_length}/t \; = 40 * 8/t$ [b/s].
- The parameters of the loss process on the network, typically p and q for the Gilbert model (please see below)

In this context, it would be useful to have 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

4.2.1 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 (see Figure 2). States 1 and 0 respectively correspond to a packet loss

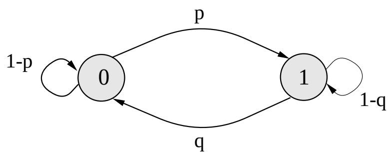

Figure 2: The Gilbert Model

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 $\pi_0^* = \frac{q}{p+q}$ $\pi_1^* = \frac{p}{p+q}$.

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 true in reality [11], we prefer to assume that the average packet loss length is 1.5 (therefore $q = 2/3$). This ambitious assumption is based on the works of [14] and [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. So p can be computed as

$$\pi_1^* = \frac{p}{p+2/3} \Leftrightarrow p = \frac{2}{3} \cdot \frac{\pi_1^*}{1-\pi_1^*}$$

4.2.2 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:

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

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:

$$P^1 = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}$$

one can compute [12] that

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

hence

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

Therefore the probability to lose both packets is:

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

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

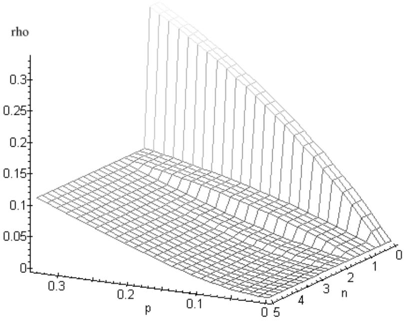

Figure 3: $\boldsymbol{\rho}$ function of $\boldsymbol{n}$ and $\boldsymbol{p}$ with $\boldsymbol{p} \in [\mathbf{0}; \mathbf{0.33}]$

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

Extrema are obtained for zeros of Equation 2. Three cases are distinguished:

- $\mathbf{0 < 1 - \boldsymbol{p} - \boldsymbol{q} < 1}$ All members of (2) 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.
- $\mathbf{1 - \boldsymbol{p} - \boldsymbol{q} = 0}$ All values of n are optimal since ρ is then constant.
- $\mathbf{-1 < 1 - \boldsymbol{p} - \boldsymbol{q} < 0}$ The logarithm function is not defined. We constat on Equation 1 that the smallest value of ρ is obtained for $n = 1$.

Conditions can be simplified in: $\boldsymbol{p} < \mathbf{1/3}$, $\boldsymbol{p} = \mathbf{1/3}$, $\boldsymbol{p} > \mathbf{1/3}$ since $\boldsymbol{q} = \mathbf{2/3}$.

Figures 3 and 4 are plots of $\boldsymbol{\rho}$ as a function of $\boldsymbol{n}$ and $\boldsymbol{p}$, first one illustrates case $\boldsymbol{p} < \mathbf{1/3}$ second one all three cases.

On figure 3 we can constat that when $\boldsymbol{n}$ increases $\boldsymbol{\rho}$ tends fast to its limit. For $\boldsymbol{n} = \mathbf{3}$, $\boldsymbol{\rho}$ already quite equals the limit. More offsetting probably brings a small reduction of $\boldsymbol{\rho}$ while it increases playout delay. But we do not precisely know which $\boldsymbol{k}$ 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 $\boldsymbol{n} = \mathbf{3}$. Nevertheless it nicely improves the loss rate after reconstruction compared to $\boldsymbol{n} = \mathbf{1}$ and the playout delay introduced seems reasonable[5].

Two last cases are clear. When $\boldsymbol{p} > \mathbf{1/3}$ one clearly choose $\boldsymbol{n} = \mathbf{1}$ since it guarantees the smallest delay and loss rate after reconstruction. For $\boldsymbol{p} = \mathbf{1/3}$ any value of $\boldsymbol{n}$ is optimal, so we choose $\boldsymbol{n} = \mathbf{1}$ 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;
```

Notice that these results are specific to this value of $\boldsymbol{q}$. In the general case, the test should be done with $\boldsymbol{p} + \boldsymbol{q}$.

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

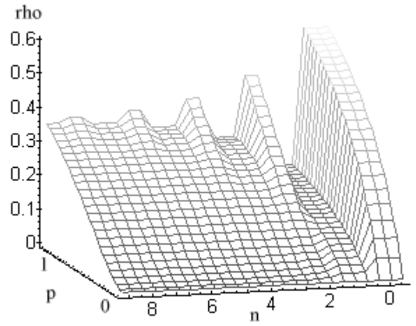

Figure 4: ρ function of n and p with $p \in [0;1]$

4.2.3 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^* (the probability to lose a packet), if redundancy is used ρ is the loss rate after reconstruction.

It is not very clear how much loss audio can be tolerated at destination, it depends on voice reconstruction techniques, audio unit's length and the subjective tolerance of user. In the software three reconstruction techniques are available: silence substitution, packet repetition and pattern matching repair. It appears [11] that with packet repetition techniques, 5% loss rate after reconstruction can be tolerated. We assume then $\tau = 5\%$. The choice is then very simple:

if ($\pi_1^* > \tau$) $k = 2$;
else $k = 1$.

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

4.2.4 Encodings

The last parameters to determine are the codecs used for encoding. We use a Free-Phone [13] inspired algorithm. Their algoritm solves a more general problem where 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[6] 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 a set that performs encoding in the range of 5.6kb/s to 64 kb/s with 8 kHz sampling frequency: LPC (5.6kb/s), GSM (13.2kb/s), G726-16 (16kb/s), G726-24 (24kb/s), G726-32 (32kb/s), G726-40 (40kb/s), and A-law (64 kb/s). It represents a quite homogeneous distribution of usable rate, 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.

Recall 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 $i = 1 : k$.

[6] the first audio unit sent

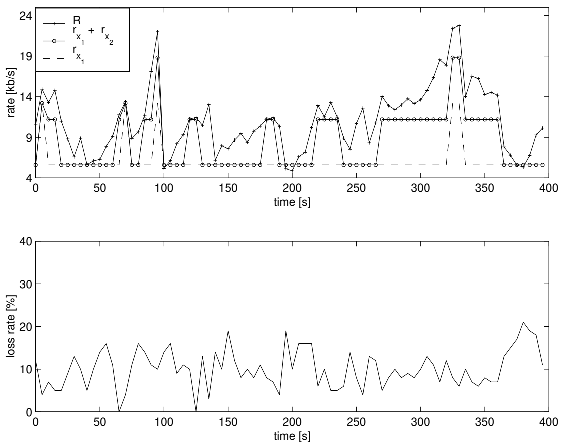

Figure 5: Source rates and network loss

```
x0 = 1; // ensures minimum encoding for(i=1:k-1:i++) xi = 0; r = r_x0;
i = 0;
do{
   if(xi < N) xi++;
   r=0;
   for(j=0; j<k; j++){
         r += r_xj;
         }
   if (r > R){
         if(x0==1 && x1==0) break;
         xi--;
         break;
         }
   if(x_{k-1} = N) break; // all codecs have the best quality
   i = ++i % k;
   }while(r < R);
```

4.2.5 Frame duration

We chose 20ms as frame duration. It enables smaller latency and reduces time to wait to receive offseted packets. Anyway, it is still possible to change this manually on the audio interface, since we did not include automatic control for this parameter[7].

5 Evaluation

5.1 Results

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

During about a 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 (Mbone).

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 second and the average loss rate above 10%. Figure 5 shows the TCP-friendly rate constraint $\boldsymbol{R}$ the payload rate $\boldsymbol{r_{x_1} + r_{r_2}}$ and the rate used by the

[7]It would be obvious to do so with ``tool.rat.rate'' command

primary encoding $\boldsymbol{r_{x_1}}$ during a 400 seconds connection. This permits to see when redundancy is applied in response to loss on the network presented on the bottom of figure.

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 mesasurements of loss rate after reconstruction here since the efficiency of the method was demonstrated by [13] and [6].

6 Implementation

This section gives some specific hints about the implementation of the adaptive channel coder algorithm. It adresses mainly people who will want to improve it.

6.1 Principle

Our background idea to change channel coder properties was to copy the Mbus messages sent by the user interface as user clicks. This enables fast and easy programming since Mbus commands are made for this task. To discover which message to send, we print the Mbus commands received by the media-engine as clicks occurs.

It is relevant to change the channel coding each time a RTCP packet is received and processed. Therefore this action is initiated at the end of the function that processes RTCP receiver reports.

6.2 Mbus commands used

Mbus messages are usually sent using UDP, but since the messages come from the media engine and are intended for himself it is not needed to send them, fuctions call be called directly. We explicitely used:

- To select primary codec:
  ''<tool.rat.codec> <primary>''
  primary is a string of form ''<Codec_name> <Stereo or Mono> <frequencykHz>''
- To select channel coder type, redundant codec, and offset:
  ''<audio.channel.coding> <secondary>''
  secondary is a string of form ''<redundancy or none> <Codec_name> <offset>''

6.3 Future work

The most important thing to do seems to build a function to estimate quality perceived at reception. Without it, choices must be statically done. If we had it, it would be possible to 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 $\boldsymbol{q = 2/3}$. 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 perform this at the present time, but it seems not too difficult 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 seem not useful.

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

7 Conclusion

In this paper we proposed a method to provide adaptive source coding for the Robust Audio Tool. It satisfies fair-rate constraints while improvig 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.

8 Acknowledgements

We would like to thank RAT hackers, Orion Hodson and Colin Perkins for their active support. For us it represented a fine opportunity to learn concepts of programming, provided by experienced programmers.

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.