ADAPTIVE *Delay aware* ERROR CONTROL FOR INTERNET TELEPHONY

Catherine Boutremans, Jean-Yves Le Boudec
Institute for Computer Communications and Applications
Swiss Federal Institute of Technology at Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{Catherine.Boutremans,Jean-Yves.LeBoudec}@epfl.ch

Abstract— **Forward Error Correction copes with packet losses, but at the expense of an increase of the end-to-end delay. By failing to take this into account, existing error control schemes for audio often lead to end-to-end delays larger than 150 ms, which has an impact on the perceived audio quality. In this paper, we develop an adaptive error control scheme for audio which is delay aware, i.e. which incorporates the impact on the end-to-end delay in the choice of FEC.**

To this end, we model the perceived audio quality as a function of the end-to-end delay and of the encoding rate at destination. We develop a joint rate/error/delay control algorithm which optimizes this measure of quality and is TCP-Friendly. We show that our scheme increases utility in a single class best effort network. We evaluate the benefit for audio sources to use the Alternative Best Effort service, which offers applications the choice between lower end-to-end delay and more overall throughput.

Keywords—**IP telephony, FEC, end-to-end delay, TCP-Friendly**

I. INTRODUCTION

Real-time, interactive audio over IP networks often suffers from packet loss. Forward Error Correction (FEC) can be used [1] to mitigate the impact of packet losses. However, FEC has the drawback of increasing the end-to-end delay. Indeed, in current systems, FEC requires the destination to wait for several packets to be received in order to repair packet losses.

When used over the standard IP best effort service, an audio source should also control its rate in order to react to network congestion and share bandwidth fairly, in some sense [2], [3]. Recently, Bolot [1] proposed an adaptive rate/error control which optimizes a subjective measure of quality and incorporates the constraint of rate control. This scheme proved to be efficient but it does not try to optimize the overall end-to-end delay; in particular, it does not manage the additional delay due to FEC.

Now, it is recognized that above a certain threshold (around 150ms) the end-to-end delay starts to be annoying [4]. In some cases, it may be that a rate/error control such as [1] causes the delay to be larger than the threshold, whereas it would have been possible to avoid this, at the expense of an increased distortion. If the resulting distortion is still acceptable, then a reduced delay would be preferable. This is our main motivation to propose a delay aware scheme for controlling rate and FEC. Our simulation examples in Section VI-A tend to indicate that our method does avoid that a source waste delay on the FEC when it is not necessary.

A secondary motivation is the emergence of a number of proposals for internet differentiated services which propose to combine performance objectives related to delay and to throughput or packet loss [5], [6], [7], [8]. The proposal in [8] is based on per flow queuing and assumes that sources employ packet pair probing. In this paper, we focus on another proposal, Alternative Best Effort (ABE) [9], which does not require per flow queuing. It offers an application the choice of either a lower end-to-end delay or more overall throughput. In today's internet, where explicit congestion notification (ECN) [10] is not yet widespread, the low delay class is likely to receive more packet loss. This asks the question whether there is a real benefit for an audio application to use the low delay class, since it may be forced to compensate the additional loss by more FEC, and thus more end-to-end delay. We show some simulation results in Section VI-B with audio sources implementing our delay aware control scheme; they tend to indicate that the answer is positive.

In this paper, we propose an adaptive error control scheme for real time audio over best effort networks which is *delay aware*, namely, which chooses the FEC according to its impact on the end-to-end delay. We start by considering the perceived audio quality as a function of the audio encoding rate received at the destination *and* of the end-to-end delay. Then we write our control problem as an optimization problem, and solve it numerically and theoretically. We also incorporate a TCP-friendly module, in order to ensure that our rate conforms with current practice for the Internet. This gives us the basis for a rate/error/delay control algorithm which (1) optimizes our delay-aware measure of audio quality and (2) is TCP-Friendly.

This paper constitutes an extension of the work reported in [1] where a rate/error control problem was first formulated as an optimization problem. However, our work brings two new contributions. First and foremost, we consider the end-to-end delay in the optimization problem. Second, we introduce a TCP-Friendly module to control the rate of the audio stream. We also provide details about the resolution of the optimization problem ([1] did not explain in details how the optimization was solved).

The validation of a particular measure of perceived audio quality is outside the scope of this paper. Therefore, we increase robustness by plugging into our method several different quality measures.

The paper is organized as follows. Section II reviews the state of the art on error control for audio applications and TCP friendly rate control. Section III describes our utility function approach. Section IV presents our main result, namely, the delay aware error and rate control. An auxiliary component for implementing TCP-friendliness is described in Section V. Results of simulations implemented in ns2 [11] are given in Section VI-A (today's internet) and Section VI-B (ABE network). Details about the optimization problem and its solution are in appendix.

II. Related work

A. Error recovery

As best-effort networks do not provide any guarantee in terms of Quality of Service, the end-points in an audio transmission must be able to recover from packet losses. To this end, an audio transport protocol may cope with packet losses by [12]: (1) retransmitting dropped packets, (2) applying Forward Error Correction (FEC) to reconstruct the missing packet, or (3) using error-concealment algorithms to correct the losses.

Retransmission algorithms based on Automatic Repeat reQuest (ARQ) have been successful in protocols like TCP, but they are typically not acceptable for real-time audio applications since they dramatically increase the end-to-end delay. FEC, on the other hand, is an attractive alternative to ARQ since it provides relatively low-delay performance. The principle of FEC is to send redundant information, along with the original information, so that lost data can be recovered, at least partially, from this redundant information.

Originally, there was much interest in the provision of *media independent* FEC using block codes (*e.g.* based on Reed-Solomon [13] or on parity codes [14], [15]) to provide redundant information. Unfortunately, these techniques have the disadvantage of introducing important additional delays (because a source must wait for the entire block of packets before computing and transmitting the redundancy packet). These schemes can therefore be used for one-way, near real-time audio transmission but are not suitable for interactive audio communications.

One way to avoid this coding delay is instead of sending redundant information rely upon *error-concealment* algorithms at the receiver to correct the effect of the missing packets [16]. The most simple error-concealment techniques replace the missing audio unit with silence, white noise or a repeated segment. As such, these techniques work well for relatively small loss rates ($\leq 10\%$) and for small packets (4-40ms of audio) but they break down when the loss length approaches the length of a phoneme (5-100ms). More sophisticated concealment techniques, such as Adaptive Packetization and Concealment (AP/C)[17], exploit the network loss characteristics and the property of long-term correlation within a speech signal together, to mitigate the impact of packet losses. This is accomplished by an adaptive choice of the packetization interval of the voice stream at the sender. Besides, modern speech codecs, such as G.729 [18], use concealment algorithms that are defined as part of their specification. These algorithms interpolate the missing codec parameters based on surrounding and previous values. These techniques lead to a good quality of the reconstructed speech if the number of consecutive lost frames is small and if the loss does not occur at an unvoiced/voiced transition [19]. Hence, error-concealment schemes should not be regarded as substitutes for FEC, but rather a combination to the latter.

Most audio conferencing tools use a *media specific* FEC scheme that combines error-concealment and FEC. The principle of the *media specific* FEC is to transmit each segment of audio, encoded with different quality coders, in multiple packets. When a packet is lost, another packet containing the same segment (maybe encoded differently) can be able to cover the loss. This approach has been advocated by Hardman *et al* [16] and Bolot *et al* [20] for use on the Mbone, and extensively simulated by Podolsky *et al*. [2].

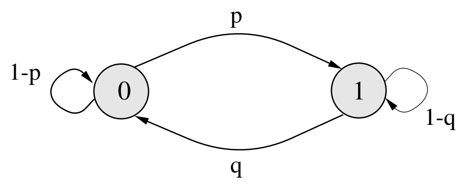


Fig. 1. *The Gilbert model*

The first transmitted copy of the audio segment is referred as primary encoding and subsequent transmissions as secondary encodings. In the *media specific* FEC scheme, redundant audio segments are piggy-backed onto a later packet which is preferable to the transmission of additional packets, as this decreases the amount of packet overhead and routing decisions. For example, in the case of a single redundant segment, packet n contains, in addition to its encoded samples, a redundant version of packet $n-1$. This redundant information is usually obtained using a lower-bit-rate, lower-quality encoding than the primary information. This simple scheme only recovers from isolated losses but can be modified (as proposed in [3]) to recover from consecutive losses as well by carrying in packet n redundant versions of packets $n-1$ and $n-2$, or of packets $n-1$, $n-2$ and $n-3$ or of packets $n-1$ and $n-3$ etc.

Obviously, the more redundant information is added at the source, the more lost packets can be reconstructed. However, sending more redundant copies implies increasing the bandwidth requirement at the source (and henceforth the packet loss rate) and increasing the end-to-end delay (since the receiver has to wait longer for the redundant information). Moreover, it would make little sense to add much redundant information when the network load is low and the packet losses are rare.

Therefore, a robust FEC scheme should be adaptive and choose the FEC according to the network characteristics (such as packet loss process, available bandwidth,...) at any given time and depending upon its impact on the end-to-end delay. In the following sections we describe the packet loss process of audio packets in the Internet and we review the rate control schemes that have been proposed in the literature.

B. Loss process of audio packets

The efficiency of the FEC depends on the characteristics of the loss process of audio packets. Typically, FEC is more efficient when the consecutive number of lost packets is small.

There has been much research efforts in the measurement and modeling of end-to-end Internet characteristics [21],[22],[23]. The main result is that the correlation structure of the loss process of audio packets can be modeled with low order Markov chains. In particular, a two-state Gilbert model was found to be an accurate model in many studies. Moreover, it was found that the distribution of the number of packets lost in a loss period is approximately geometric [20],[23]. These results confirmed that FEC schemes are well suited for interactive audio applications in the Internet. In the rest of the paper, we will use the

Gilbert model to characterize the loss process of audio packets. The Gilbert model (depicted in Figure 1) is a two-state model in which state 1 represents a packet loss and state 0 represents a packet reaching the destination. The parameters p and q denote respectively the probabilities of passing from state 0 (no loss) to state 1 (loss) and from state 1 to state 0. In absence of redundant information, one can easily derive the packet loss rate $PLR = \frac{p}{p+q}$. This model also allows to compute the packet loss rates after reconstruction when FEC is used. For example, if we consider the case where packet n only includes redundant information about packet $n-1$, the packet loss rate after reconstruction is equal to $\frac{p(1-q)}{p+q}$.

C. Rate control

As we mentioned earlier, the addition of FEC repair data to a media stream is an effective means by which that stream may be protected against packet loss. However, the addition of large amounts of repair data when loss is detected will increase network congestion and hence packet loss, leading to a worsening of the problem which the use of FEC was intended to solve. Therefore, the rate of audio sources should be controlled in order to avoid congestion collapse in the network.

In addition, it has been suggested that audio applications share resources fairly with each other and with current TCP-based applications. One way to ensure this 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* [24] - 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 as a TCP connection.

There has been significant previous research on TCP-Friendly control mechanisms and many control schemes were proposed in the literature. We can distinguish three main classes of control mechanisms: (1) the window-based control mechanisms, (2) the mechanisms based on additive increase, multiplicative decrease (AIMD), and (3) the equation-based mechanisms.

The control mechanisms closest to TCP are the window-based mechanisms [25], [26]. They maintain a *congestion window* which is used to control the transmission of the packets. Although window-based schemes exhibit a behavior very close to the one of TCP, their main disadvantage is their lack of flexibility. Since these protocols strictly adhere to TCP window dynamics, it would be hard to modify them to take into account timeliness requirements of real-time streams.

Another class of control mechanisms uses additive increase, multiplicative decrease (AIMD) in some form, but do not apply AIMD to a congestion window. The RAP protocol (Rate Adaptation Protocol) [27] employs an AIMD rate control algorithm based on regular acknowledgments sent by the receiver. Another AIMD protocol has been proposed in [28]. This scheme relies on regular RTP/RTCP [29] reports sent between sender and receiver to estimate the loss rates and round-trip times. While AIMD schemes exhibit good response to transient changes in congestion, real-time streams find decreasing the sending rate in multiplicative order in response to a single loss to be unnecessarily severe (as it can noticeably decrease the user-perceived quality [30]). For this reason, equation-based control mechanism seem to be leading candidates for mechanisms to provide relatively smooth congestion control for real-time traffic.

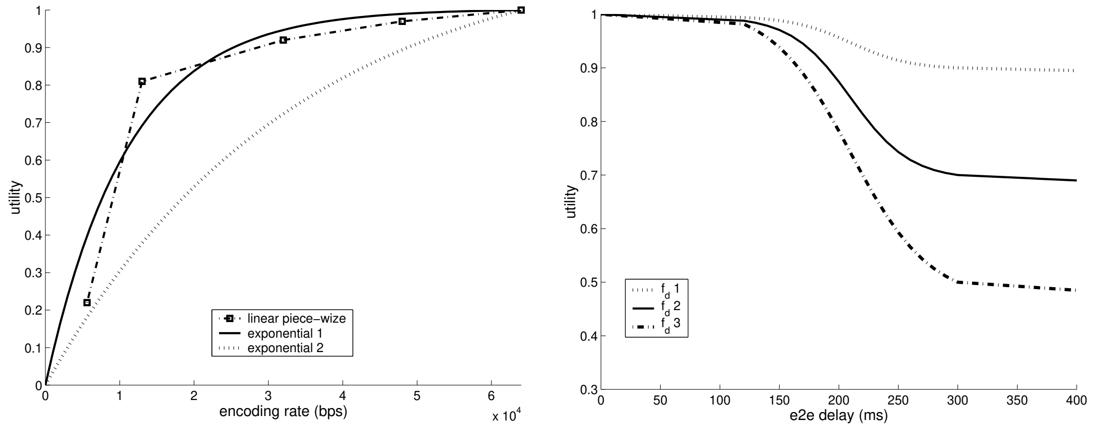
In Equation-based congestion control [24] schemes, the sender uses an equation characterizing the allowed sending rate of a TCP connection as a function of the RTT and packet loss rate, and adjusts its sending rate according to those measured parameters. A key issue, when using these schemes, is of course to choose a reliable characterization of the TCP throughput. The TCP-Friendly protocols proposed in [30], [31] are based on the TCP response function first reported in [24] and later formalized in [32]. Since this characterization does not take the *timeouts* into account, it has been reported in [32] that this model is not accurate for loss rates higher than 5%. Another formulation of the TCP response function was derived in [33]. It states that the average throughput (in bytes/sec) of a TCP connection (r_{TCP}) is given by:

$$r_{TCP} = \frac{s}{t_{RTT}\sqrt{\frac{2l}{3}} + t_{RTO}(3\sqrt{\frac{3l}{8}})l(1+32l^2)} \tag{1}$$

where s is the packet size, t_{RTT} is the round trip time, t_{RTO} is the TCP re-transmission timeout and l is the frequency of loss indications per packet sent. Note that l is not exactly equal to the packet loss rate but the latter provides an upper bound on the value of l and can be used as an approximation. Based on this model, Padhye and al. [33] proposed a scheme in which the receiver acknowledges each packet. At fixed time intervals, the sender estimates the packet loss rate experienced during the previous interval and updates the sending rate using equation (1). Since this scheme updates the sending rate at fixed time intervals, it is suitable for use with multimedia applications. But it has the disadvantage to have a poor transient response at small time-scales. Besides, Floyd and al. recently proposed the TFRC protocol [34]. In TFRC, the sender explicitly adjusts its sending rate as a function of the measured rate of *loss events*, where a loss event consists of one or more packets dropped within a single RTT. Their algorithm for calculating the loss event rate is based on the method of Average Loss Interval. It offers a very good tradeoff between responsiveness to changes in congestion and avoidance of unnecessary abrupt shifts in the sending rate. Unfortunately, TFRC cannot be used, as such, with audio streams for two main reasons: first, because it requires the receiver to send feedback to the source at least once every RTT, which is not recommended when using RTCP for feedback collection; second, because the method of Average Loss Interval is not appropriate in the case where sources adjust their sending rate by changing the packet size while keeping the interval between packets constant. Our rate control scheme, which will be described in section V, was adapted from the work of Padhye ([33]) and adjusts the rate by keeping the interval between packets constant.

III. A UTILITY FUNCTION APPROACH WHICH ACCOUNTS FOR DELAY

As mentioned above, the purpose of our work is to design an error control algorithm which is *delay aware*, namely, which incorporates the impact on the end-to-end delay in the choice of the FEC. To this end, we take a utility function approach.



(a) (b)

Fig. 2. *Utility as a function of (a) the rate only (for delay=0), (b) the end-to-end delay only (for encoding rate = 64Kbit/s)*

For users employing our audio application, we define the utility not only in terms of sound quality at the destination but also in terms of *interactivity*. In other words, we consider the utility (quality) as being a function of the encoding rate of audio received at the destination *and of the end-to-end delay*.

This utility function $f : \Re^+ \times \Re^+ \rightarrow [0, 1]$ was obtained as follows. We characterized separately:

• **the utility as a function of the encoding rate**: $f_r : \Re^+ \rightarrow [0, 1]$. There exists objective and subjective methods to assess the quality of an audio source as a function of the encoding rate. Objective methods use specific signal metrics to assess the quality, such signal-to-noise ration (SNR) [35]. These metrics, while easy to obtain, are only approximations for two main reasons. First, because they are sensitive to the characteristics of the signal, and hence to the words being pronounced. Second, because they often fail to correlate with perception properties of the human hearing system [36]. Despite these limitations, a SNR model can still give insight into the audio quality. Moreover, operational SNR curves (on a linear scale from 0 to 1) can be modeled using negative exponentials, the quality increasing rapidly at low rates, and more moderately at higher rates. Therefore, we considered two utility functions of the form: $f_r(x) = c_1 + c_2\, e^{-\alpha x}$, where c_1 and c_2 are constants, selected so that $f_r(64Kbit/s) = 1$ and $f_r(0) = 0$. Figure 2(a) depicts two of these exponentials, corresponding respectively to $\alpha = 0.0001$ and $\alpha = 0.00003$. Quality assessment models based on subjective measurements provide more accurate results but are more difficult to obtain. A widely used model is MOS (Mean Opinion Scores) where the perceived quality is usually rated on a 1 to 5 scale. Utility functions for adaptive flows can be obtained using score tables (scaled down between 0 and 1), and interpolating between values to obtain piece-wize linear utility functions. We present such a function in Figure 2(a). Besides, exponentially-decay functions were introduced in [37] to describe utility curves for adaptive application. These curves account for the fact that the quality increases slowly at very low rates (below the minimum rate required by the application), then rapidly at intermediate rates and again slowly at higher rates. In this work, we restricted ourselves to consider strictly exponential curves, but more complex functions could be used for future work.

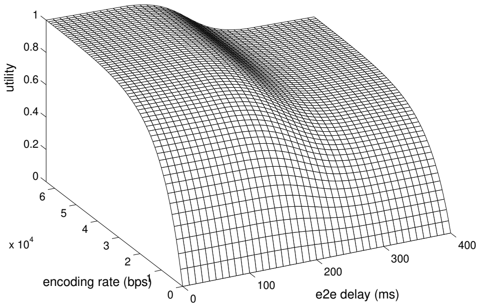


Fig. 3. *Utility as a function of the rate and the end-to-end delay*

• **the utility as a function of the end-to-end delay**: $f_d : \Re^+ \rightarrow [0, 1]$. Even though an objective and unique function can not be set to describe the quality as a function of the end-to-end delay, various studies concluded that, for natural hearing, the end-to-end delay should be approximately 150ms [4], [38]. While a lower delay can not really be appreciated, delays above this threshold will be noticed by the users and will become a hindrance to interactivity. Moreover, it is also recognized that telephony users find delay of greater than about 300ms more like half duplex connection than a conversation. These considerations lead us to consider utility curves like those represented on figure 2(b). These utility functions present the following behaviour: for delays below the critical threshold of 150ms, the quality decreases very slowly as the users do not benefit from getting a lower end-to-end delay. Then, above this threshold, the quality drops steeply as any increase of the end-to-end delay hurts the interactivity. Then, above 300ms, since the connection is considered as a half duplex conversation anyway, any further increase of the delay only slightly affects the quality (which is already low). Even though it agrees with common intuition, the nature of the exact behaviour of this function remains out of the scope of this study. Our goal is not to validate a particular utility function but rather to show that the use of this kind of function allows to incorporate the impact on the end-to-end delay in the choice of FEC. To this end, we considered a set of utility functions of the form:

$$f_d(x) = \begin{cases} 1 - \gamma_1 x & \text{if } x \leq 150 \\ b_1 \tanh(\beta(x - b_2)) + b_3 & \text{if } 150 < x < 300 \\ 1 - \delta - \gamma_2 x & \text{if } x \geq 300 \end{cases}$$

where x is the end-to-end delay (in ms), γ_1, β and γ_2 are parameters representing the steepness of the decrease in each of the 3 regions and δ determines the difference between the full and the half-duplex quality. b_1, b_2 and b_3 are constants selected to ensure the continuity of f_d. The utility functions depicted on figure 2(b) were obtained by tuning these parameters. A user will opt for either one or the other depending on the importance she attaches to the end-to-end delay.

Then, the global utility is defined as the product of f_d and f_r:

$$f(x,y) = f_r(x)\, f_d(y)$$

where x and y represent respectively the reconstructed rate at the destination and the end-to-end delay. Such a function is depicted on figure III.

Note that the packet loss rate after reconstruction also impacts the quality. Many codecs employ various concealment techniques which are able to mask such losses as long as $PLR_{after\, reconstr} \leq PLR_{max}$ where the threshold PLR_{max} depends on the codec. We could have incorporated this in our utility function approach. However, for tractability, we have chose to express this as a constraint, namely, we accept only encodings such that $PLR_{afterreconstr} \leq PLR_{max}$. Whether there is value in incorporating $PLR_{afterreconstr}$ in the utility function, and how to do it, is the object of future research.

In the simulation we have performed, we have used various values for the parameters of the utility functions.

IV. Our joint rate/errror/delay control

Once we have determined the influence of the end-to-end delay on the quality (utility) of the call, our goal is to **find the best redundant information that will maximize the perceived quality at the receiver**. This is the purpose of this section.

Consider a source with the flexibility to encode its samples at a rate $x \in [0, X_{max}]$ (we suppose X_{max} to be 64 Kbits/s). The quality of the voice call is characterized by a function $f : \Re^+ \times \Re^+ \rightarrow [0,1]$ of (1) the reconstructed rate at the destination and (2) the end-to-end delay.

The source transmits voice packets to a destination over an unreliable network characterized by:

- **a packet loss process:** Y_i which we suppose to be a Gilbert process where $Y_i \in \{0,1\}$ (see section II-B). If the ith packet is received at the destination, then $Y_i = 0$, otherwise, $Y_i = 1$. The parameters p and q of the Gilbert model are estimated on-line at the receiver using the maximum likelihood estimator.
- **a delay distribution.** We don't make any assumption about the distribution of delays in the network. Rather do we consider the 99 percentile of the delays experienced by the voice packets, which we call d_{net}. d_{net} actually represents the playout delay of the voice packets and is estimated at the destination as described in [39].

The parameters p, q and d_{net} (which are all estimated on-line at the receiver) are sent back to the source via the application specific part (APP) of the RTCP receiver reports.

Let R_{max} be the rate available for the audio flow. R_{max} is the result of our TCP-Friendly rate control scheme (which is described in the next section) and is updated upon reception of an RTCP receiver report.

Then, consider that we use the *media specific* FEC scheme described above to recover from packet losses. Let $K-1$ denote the maximum number of redundant pieces of information sent along with the primary information. Thus, packet n carries information about at most (i.e. a subset of) packets $n-1, \ldots, n-K+1$. Therefore the total number of copies (encoded at different rates, including 0) of a given packet sent by the source is equal to K. The optimal value of K is a priori unknown and is supposed to be in $[1, K_{max}]$. In practice, the larger K, the longer the destination has to wait to receive the redundant information, and thus, the longer the end-to-end delay. Let τ_K represent the delay introduced by the use of FEC. τ_K is the delay between sending the first and the last copy of a given packet and can thus be written as follows: $\tau_K = (K-1)\, pcktInt$, where $pcktInt$ is the time interval between two consecutive audio packets.

Further, define the random variable Δ to be $\Delta = \{i | Y_i = 0, i = 1, \ldots, K\}$, namely the set of copies of a given packet that are received at the destination.

Then, our problem can be stated as follows: Given that we can send at most K_{max} copies of each voice packets, find the optimal number of copies to send, and the optimal encoding rate for each copy, so as to maximize the quality of the voice call subject to the rate constraint. Mathematically, it gives the following optimization problem (which we call P1):

$$\underset{\substack{1 \leq K \leq K_{max} \\ x_i\,(i=1,\ldots,K)}}{\text{maximize}} \quad \sum_{\Delta \subseteq \{1,\ldots,K\}} P(\Delta) \max_{i \in \Delta} f(x_i, d_{net} + \tau_K)$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{K} x_i + R_{overhead} \leq R_{max} \\ x_i \geq 0, i = 1, \ldots, K \\ PLR_{after\, reconstr} \leq PLR_{max} \end{cases}$$

where x_i is the encoding rate of the copy placed in ith position in the stream ($i = 1$ corresponds to the primary information); $P(\Delta)$ is the probability to receive the set Δ; $R_{overhead}$ is the bandwidth overhead of the IP/UDP/RTP headers, $PLR_{after\, reconstr}$ is the packet loss rate after reconstruction and PLR_{max} is the maximum acceptable value for $PLR_{after\, reconstr}$. The choice of a value for PLR_{max} mainly depends on the efficiency of the error resilience scheme used at the receiver. Typical values of PLR_{max} range between 5 and 10% [40].

The objective function above represents the average quality measured at the destination. This model assumes that different copies of a given packet can not be combined to produce a better quality copy of the original packet. We rather assume that the receiver will send to its audio driver the *best* copy (i.e. leading to the highest quality) it has received of a given packet. The formulation of the objective function could be different if we used layered or multiple description coding [41] schemes for audio. But this is kept for future work.

The problem above appears to be, in general, difficult to solve but a careful analysis of the objective function allowed us to derive solutions for K_{max} up to 5. The methodology remains the same for greater values of K_{max} but the main burden is that the number of terms in the sum grows exponentially with K, and there is no generic formulation to express the probabilities associated to each term as a function of K. For further details about the resolution of the optimization problem, one can refer to appendix A. In the following, we just describe the general characteristics of the results:

- $\boldsymbol{x_1 \geq x_i}$, $\forall i = 1, ..., K$: the primary information should always be encoded using the best quality encoding among those used to encode the different copies.
- **if** $\boldsymbol{p + q < 1}$, $\boldsymbol{x_1 \geq x_K \geq x_i}$, $\forall i = 1, \ldots, K$: it pays to use good quality coders to encode the end packets. Furthermore, for a given number of copies to send, the larger K, the better the audio quality at the destination, but also the larger the end-to-end delay. In this case, our algorithm allows to find the good tradeoff between quality of the reconstructed signal and delay.
- **if** $\boldsymbol{p + q > 1}$, $\boldsymbol{x_1 \geq x_2 \geq \ldots \geq x_K}$, $\forall i = 1, \ldots, K$: the redundant copies should closely follow the primary packet and the quality of the encodings should decrease as the copies are moved away from the primary packet.

V. THE TCP-FRIENDLY RATE CONTROL MODULE

Our TCP-Friendly rate control relies on the Real-Time Transport Protocol (RTP) [29] and its control part, RTCP. In our scheme, the sender performs equation-based congestion control based on feedback information contained in RTCP reports. In details, it works as follows: about every 5s (as advised in [29]), the receiver issues an RTCP report containing the packet loss rate experienced since sending the last report and various timestamps. After receiving the nth RTCP receiver report, the sender estimates the bandwidth share it should be using as follows:

- It computes the round-trip-time, t_{RTT_n} using the timestamps contained in this report and updates the TCP timeout value (t_{RTO_n}) accordingly.
- To avoid reactions to sudden loss peaks in the network, it maintains an Exponentially Weighted Moving Average of the packet loss rate:

$$PLR_n = (1 - \alpha)\, PLR_{n-1} + \alpha\, PLR \quad (2)$$

where PLR is the packet loss rate reported in the RTCP packet and α is the smoothing factor set here to 0.3. This value was also suggested in [42].

- Then, it estimates the TCP-Friendly rate r_{TCP_n} (in bytes/sec) using equation (1) with the parameters t_{RTT_n}, t_{RTO_n} and PLR_n. The value of s is set to the packet size of a competing TCP connection. Typically, s can be set to 576 Bytes (MSS) or 1500 Bytes (MTU).
- And finally, it updates the sending rate (r_n) as follows:

```
if    (r_TCPn < r_n-1)
  decrease sending rate to r_TCPn
else
  r_AI = r_n-1 + sT / t_RTTn^2
  increase sending rate to min[r_TCPn, r_AI]
```

where $T \sim 5s$ is the time elapsed since receiving the last RTCP report and r_{AI} is the rate that a TCP connection would have reached by increasing its congestion window by one packet every round trip time. This last condition was introduced to make sure that our audio flow does not increase its bandwidth share faster than a TCP connection sharing the same link.

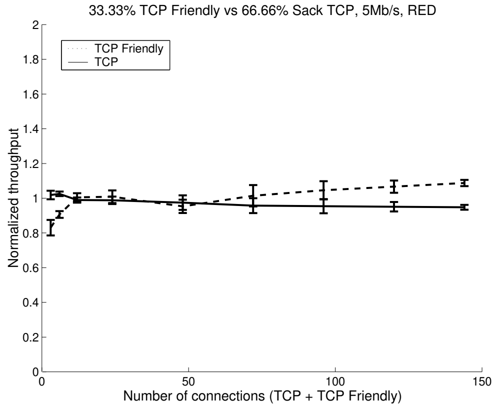


Fig. 4. *Our rate control protocol competing with TCP.*

The audio source then adjusts its sending rate by **changing the packet size**, the time interval between packets remaining constant. Usually, audio packets are generated every 20, 40 or 80ms. In this work, we fixed this interval to 20ms.

We have tested our rate control protocol in the ns [11] simulator. Figure 4 illustrates the fairness of our protocol when competing with TCP Sack traffic in a RED queue. In these simulations, n TCP Friendly and $2n$ TCP flows share a common bottleneck. Graphs show the mean throughput of each type of flow, averaged over the last 300s of simulation and over all the connections, and normalized so that a value of 1 corresponds to a fair share of the bandwidth. Each point on this graph represents the results averaged over 5 simulation runs and the corresponding confidence intervals. This figure shows that our rate control protocol co-exists fairly with TCP under a wide range of network conditions.

VI. SIMULATION EXAMPLES

We implemented and tested our *delay aware* error control scheme in the ns2 simulator [11]. This section presents a summary of our results. Two types of IP networks were considered: (1) Single Class Best-Effort networks (that we call 'Flat' networks) and (2) Alternative Best-Effort networks.

A. Flat Best-Effort Networks

This section investigates the behaviour of our scheme under a wide range of loss conditions. We consider a simple scenario where n audio (TCP-Friendly) flows and $3n$ Sack TCP flows share a single bottleneck link. The packet loss rate experienced by the connections is varied artificially by changing the number of connections sharing the link. Figures 5(a) and (b) represent respectively the packet loss rate experienced by audio flows and the corresponding TCP-Friendly rate constraint as a function of the number of connections sharing the link. The bottleneck bandwidth is 5Mbits/s and the one-way propagation delay (without queuing) is the same for all connections and is fixed at 100ms. The graphs show the mean values averaged over the last 300s of simulation and over all connections. Each point on this graph represents the results averaged over 5 simulation runs and the corresponding 99% confidence intervals. For our experiments, the parameter PLR_{max} was set to 5%.

As explained in Section III, various values can be used for the parameters of the utility function $f = f_r f_d$. Figure 5(c) shows the mean utility obtained by the sources for three different parameter settings of f_r, for a fixed $f_d = f_{d2}$ (see Figure 2(b)). The results obtained with the piecewize linear f_r (MOS) and with the exponential of parameter $\alpha = 0.0001$ (exponential 1 on the figure) were extremely close to each other and the results obtained with the exponential of parameter $\alpha = 0.00003$ differed slightly in the way they spread the rate among the different copies. In all cases, the end-to-end delay (including FEC) was the same. In the rest of the paper, all the results shown were obtained using the exponential of parameter $\alpha = 0.0001$ for f_r. We now consider three different utility functions, which are defined as follows: Utility i $= f_r f_{d_i}$, for $i \in \{1, 2, 3\}$ where the functions f_{d_i} are the one depicted in Figure 2(b). Utility 1 corresponds to a source that does not attach much importance to the delay. Utility 3 characterizes a source for which delay is an important issue and Utility 2 represents a tradeoff between delay and audio distortion issues.

Figures 5 (d), (e) and (f) compare the mean utility obtained when using the delay aware FEC to the one obtained when using the FEC scheme proposed in [31] (which we will refer to as *classical FEC*) for the three utility functions of interest. For clarity, we do not show confidence intervals but we just mention that they were small ($< 2\%$). On these figures, each curve corresponds to a particular parameter setting (i.e. value of K) of the classical FEC scheme. As one can see, there is no optimal setting of the classical FEC which maximizes the utility in all loss conditions. The **delay aware scheme** (bold curve on the figures), in return, **always chooses the optimal amount of FEC (i.e. yielding the maximal utility) depending on network conditions**. Figures 5 (g), (h) and (i) show the corresponding end-to-end delays (including the FEC-induced delay) obtained with each scheme. One can observe that, in the delay aware scheme, the end-to-end delay is adapted, depending on loss conditions. The delay is kept small when the losses are moderate and is increased when the losses become more significant. Moreover, when comparing figures 5 (g), (h) and (j), one can see that, depending on the importance the user attaches to the end-to-end delay, the amount of redundancy used is increased more or less rapidly as the loss rate increases. The same simulations were performed with smaller values of the propagation delay and showed that, in the case where this delay is very small (i.e. 50 to 80ms), the delay aware scheme leads to performances similar to the classical FEC with parameter $K = 3$ or $K = 4$, which could have been expected since, in this case, the delay induced by the FEC has no consequence on the perceived quality (because we stay below the critical threshold of 150ms).

In the light of these results, we can conclude that the delay aware scheme increases the utility by avoiding a source wasting delay using FEC when it is not necessary (and when it could hurt the perceived quality).

B. Alternative Best-Effort Networks

As stated above, a secondary motivation to develop a delay aware scheme is the existence of a number of proposals for internet differentiated services which propose a tradeoff between delay and throughput [8], [7]. In the case of TCP-Friendly audio sources whose sending rate is controlled via packet losses, this tradeoff is not simple. Indeed, in today's Internet, a source that chooses a low delay class will be likely to experience more losses and hence may be forced to compensate this additional loss by more FEC, and thus more end-to-end delay. This raises the following question: **does an audio source benefit from trading delay for throughput?**

In this paper, we chose to look into the proposal for ABE service [9], which is closer to today's Internet context. We leave the analysis of [8] for further study. With ABE, every best-effort packet is marked either *green* or *blue*. Green packets are guaranteed a low bounded delay at every router, but, in return, are more likely to be dropped during periods of congestion than blue ones. As a consequence, they will typically receive less throughput than blue ones. In this framework, the question above can be reformulated as follows: is it worth being green? In the sequel, we make use of our delay aware scheme to provide an answer to this question.

We considered a topology consisting of 2 consecutive bottleneck links. n Long flows traverse both bottlenecks, n small flows traverse only the first link and n small flows traverse only the second link. The adaptive audio applications represent one third of the connections of each type, the other two thirds are Sack TCP connections. Among the adaptive applications, half are green (i.e. use only green packets) and half are blue. Sack TCP connections use blue packets. Other flow repartitions, which are not covered here, were simulated and lead us to similar conclusions to the one presented here.

Let us first consider the case of small flows. Figures 6 (a) and (b) show the packet loss rates and the corresponding TCP-Friendly rate constraint for small green and blue flows. The bottleneck bandwidth is 5Mbits/s and the propagation delay is 50ms. Figure 6 (c) depicts the 99% percentile of the end-to-end delay (non including the FEC) experienced by green and blue packets. The tradeoff of delay and loss (or equivalently throughput) appears clearly on these three figures. Figures 6 (d), (e) and (f) present the utility received by the flows for the three utility functions of interest: Utility 1, 2 and 3 (see description above) respectively. From these graphs, one can see that, when the delays are small, the difference between the utilities of blue and green flows is minor. It is even more visible in the case where Utility 2 (which represents a tradeoff between delay and distortion of audio at destination) is used.

Secondly, we consider large flows. The propagation delay is now 90ms. Figures 6 (g), (h) and (i) show respectively the packet loss rate, TCP-Friendly rate constraint and the 99% percentile of the delays in the network for green and blue flows. The utility values obtained in these cases are depicted in Figures 6 (j), (k) and (l), for the three utility functions of interest. In this case, one can notice that the difference between the utilities received by green and blue flows is much higher. Furthermore, figures 6 (k) and (l) clearly show that, a **user who attaches at least a some small importance to the end-to-end delay will**

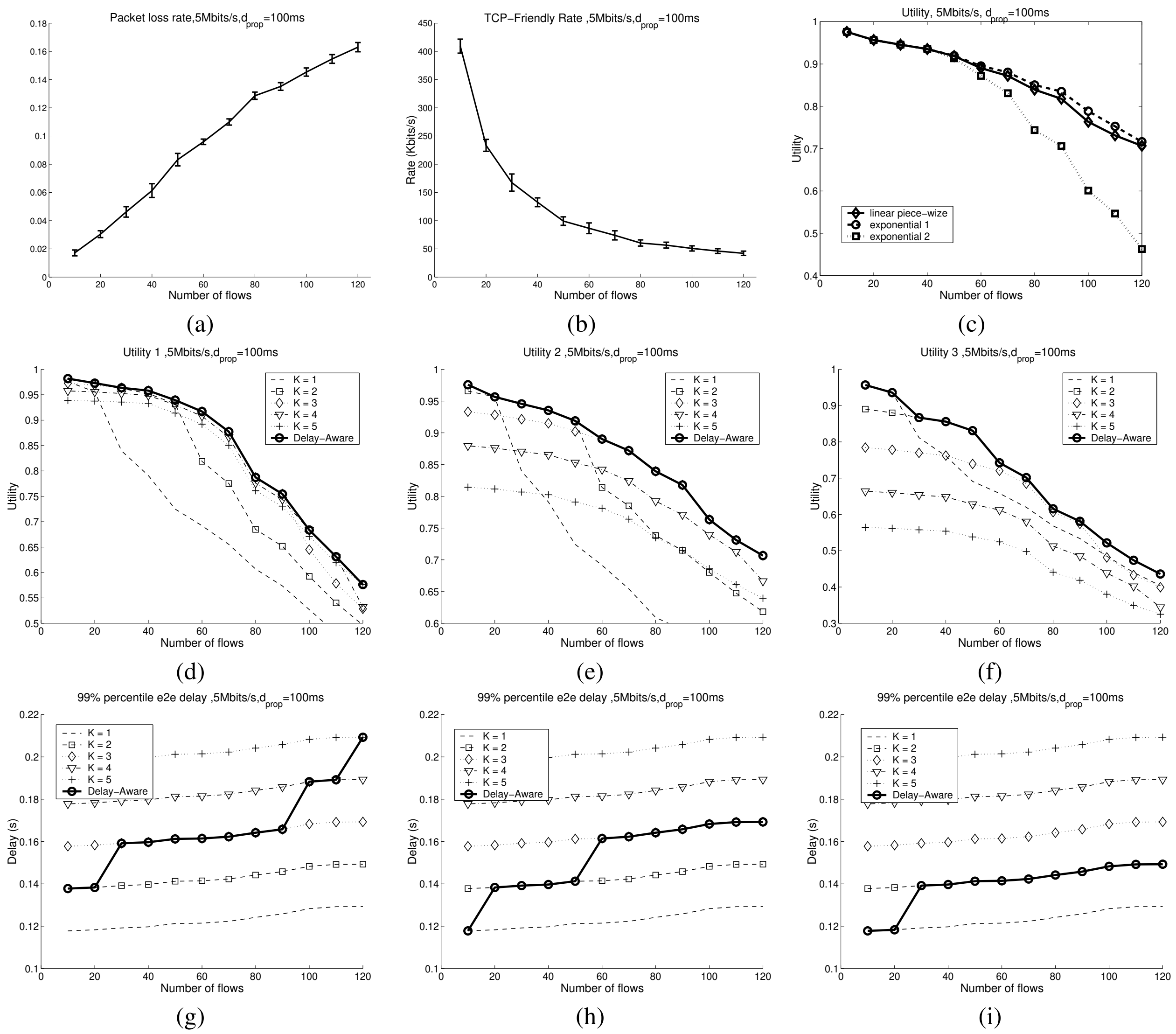


Fig. 5. *Packet loss rate (a). TCP-Friendly rate (b). Utility for different* f_r *(c). Utility of delay aware FEC vs classical FEC for (d)* $f_d = f_{d1}$*, (e)* $f_d = f_{d2}$*, (f)* $f_d = f_{d3}$*. e2e delay (including FEC) of delay aware FEC vs classical FEC for (g)* $f_d = f_{d1}$ *, (h)* $f_d = f_{d2}$ *, (i)* $f_d = f_{d3}$*.*

benefit from being green, up to a certain level of network congestion where the available rate becomes an impediment and the source could choose to switch to being blue in order to increase its throughput. It is up to the source to determine when it is better to switch from one color to another. This result tends to indicate that there is a need for color choosing algorithm for audio sources using ABE. On the other hand, a user who does not care about delay (see figure 6(j)), will probably choose the blue service in all cases.

From these results, we conclude that (time sensitive) audio applications benefit, when using ABE, from being green except when the network is very badly congested or in trivial cases when the delay is extremely small anyway.

VII. Conclusions

In this paper, we presented an adaptive error control scheme for audio which is *delay aware*, namely, which incorporates the impact on the end-to-end delay in the choice of the FEC. To this end, we took a utility function approach and we defined the quality as being a function of the encoding rate received at destination *and* of the end-to-end delay. We formalized our control problem as an optimization problem and solved it numerically and theoretically.

We showed by simulation that the delay aware scheme does avoid that a source waste delay using FEC when it is not necessary. Moreover, using our scheme in the framework of Alternative Best Effort networks, we showed that there is a real benefit, for audio applications, to use the low delay class of the service.

We are pursuing this work in several directions. One is to implement the delay aware scheme in a real audio software (such as the Robust Audio Tool [43]). Another is to develop Color Choosing algorithms for audio sources using the ABE service. Yet another one is to use the same utility function approach with other audio coding schemes such as multiple description coding [41].

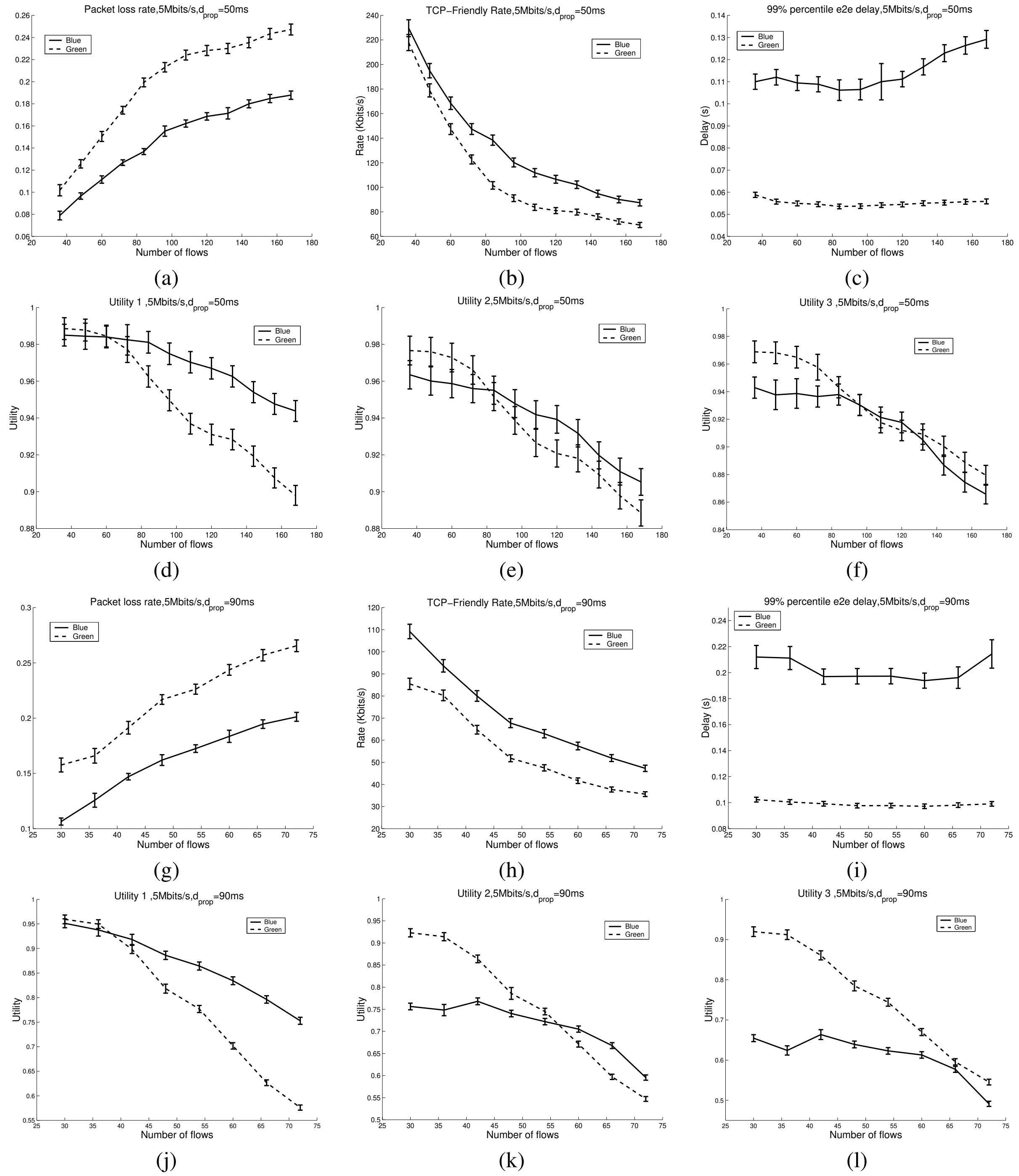


Fig. 6. *Small flows: Packet loss rate (a). TCP-Friendly rate (b). 99% percentile of e2e delay (not including FEC) (c). Utility for (d) $f_d = f_{d1}$, (e) $f_d = f_{d2}$, (f) $f_d = f_{d3}$. Long flows: Packet loss rate (g). TCP-Friendly rate (h). 99% percentile of e2e delay (without FEC) (i). Utility for (j) $f_d = f_{d1}$, (k) $f_d = f_{d2}$, (l) $f_d = f_{d3}$.*

APPENDIX

I. THE OPTIMIZATION PROBLEM

In this section, we describe the method used to solve the problem P1, which we can rewrite as follows:

$$\underset{\substack{1 \le K \le K_{max} \\ \underline{x}^K = (x_1, \ldots, x_K) \in [0, X_{max}]^K}}{\text{maximize}} \qquad F_K(K, \underline{x}^K)$$

$$\text{subject to} \quad \begin{cases} \displaystyle\sum_{i=1}^{K} x_i + R_{overhead} \le R_{max} \\ PLR_{after\,reconstr} \le PLR_{max} \end{cases}$$

with $F_K(K, \underline{x}^K) = \sum_{\Delta \subseteq \{1,\ldots,K\}} P(\Delta) \max_{i \in \Delta} f(x_i, d_{net} + \tau_K)$. For clarity, we use the notations $F_K(K, \underline{x}^K) = F_K$ and $f(x_i, d_{net} + \tau_K) = f_{i,K}$ when the context permits. In the following, we give the details for $K_{max} = 5$. We have:

$$
\begin{aligned}
F_1 &= \frac{q}{p+q} f_{1,1} \\
F_2 &= \frac{q}{p+q}(1-p) \max_{i\in\{1,2\}} f_{i,2} + \frac{p\,q}{p+q} f_{1,2} + \frac{p\,q}{p+q} f_{2,2}
\end{aligned}
$$

$$
\begin{aligned}
F_3 = {} & \frac{q}{p+q}(1-p)^2 \max_{i\in\{1,2,3\}} f_{i,3} + \frac{p\,q}{p+q}(1-p) \max_{i\in\{1,2\}} f_{i,3} + \\
& \frac{p\,q^2}{p+q} \max_{i\in\{1,3\}} f_{i,3} + \frac{p\,q}{p+q}(1-p) \max_{i\in\{2,3\}} f_{i,3} + \frac{p\,q}{p+q}(1-q) f_{1,3} + \\
& \frac{p^2\,q}{p+q} f_{2,3} + \frac{p\,q}{p+q}(1-q) f_{3,3}
\end{aligned}
$$

$$
\begin{aligned}
F_4 = {} & \frac{1}{p+q} * \{q(1-p)^3 \max_{i\in\{1,2,3,4\}} f_{i,4} + p\,q(1-p)^2 \max_{i\in\{1,2,3\}} f_{i,4} + \\
& p\,q^2(1-p) \max_{i\in\{1,2,4\}} f_{i,4} + p\,q^2(1-p) \max_{i\in\{1,3,4\}} f_{i,4} + p\,q(1- \\
& p)^2 \max_{i\in\{2,3,4\}} f_{i,4} + p\,q(1-p)(1-q) \max_{i\in\{1,2\}} f_{i,4} + p^2\,q^2 \max_{i\in\{1,3\}} f_{i,4} + \\
& p\,q^2(1-q) \max_{i\in\{1,4\}} f_{i,4} + p^2\,q(1-p) \max_{i\in\{2,3\}} f_{i,4} + p^2\,q^2 \max_{i\in\{2,4\}} f_{i,4} + \\
& p\,q(1-p)(1-q) \max_{i\in\{3,4\}} f_{i,4} + p\,q(1-q)^2 f_{1,4} + p^2\,q(1-q) f_{2,4} + \\
& p^2\,q(1-q) f_{3,4} + p\,q(1-q)^2 f_{4,4}\}
\end{aligned}
$$

$$
\begin{aligned}
F_5 = {} & \frac{1}{p+q} * \{q(1-p)^4 \max_{i\in\{1,2,3,4,5\}} f_{i,5} + p\,q(1-p)^3 \max_{i\in\{1,2,3,4\}} f_{i,5} + \\
& p\,q^2(1-p)^2 \max_{i\in\{1,2,3,5\}} f_{i,5} + p\,q^2(1-p)^2 \max_{i\in\{1,2,4,5\}} f_{i,5} + p\,q^2(1- \\
& p)^2 \max_{i\in\{1,3,4,5\}} f_{i,5} + p\,q(1-p)^3 \max_{i\in\{2,3,4,5\}} f_{i,5} + p\,q(1-p)^2(1- \\
& q) \max_{i\in\{1,2,3\}} f_{i,5} + p^2\,q^2(1-p) \max_{i\in\{1,2,4\}} f_{i,5} + p\,q^2(1-p)(1- \\
& q) \max_{i\in\{1,2,5\}} f_{i,5} + p^2\,q^2(1-p) \max_{i\in\{1,3,4\}} f_{i,5} + p^2\,q^3 \max_{i\in\{1,3,5\}} f_{i,5} + \\
& p\,q^2(1-p)(1-q) \max_{i\in\{1,4,5\}} f_{i,5} + p^2\,q(1-p)^2 \max_{i\in\{2,3,4\}} f_{i,5} + \\
& p^2\,q^2(1-p) \max_{i\in\{2,4,5\}} f_{i,5} + p^2\,q^2(1-p) \max_{i\in\{2,3,5\}} f_{i,5} + p\,q(1- \\
& p)^2(1-q) \max_{i\in\{3,4,5\}} f_{i,5} + p\,q(1-p)(1-q)^2 \max_{i\in\{1,2\}} f_{i,5} + p^2\,q^2(1- \\
& q) \max_{i\in\{1,3\}} f_{i,5} + p^2\,q^2(1-q) \max_{i\in\{1,4\}} f_{i,5} + p\,q^2(1-q)^2 \max_{i\in\{1,5\}} f_{i,5} + \\
& p^2\,q(1-p)(1-q) \max_{i\in\{2,3\}} f_{i,5} + p^3\,q^2 \max_{i\in\{2,4\}} f_{i,5} + p^2\,q^2(1- \\
& q) \max_{i\in\{2,5\}} f_{i,5} + p^2\,q(1-p)(1-q) \max_{i\in\{3,4\}} f_{i,5} + p^2\,q^2(1- \\
& q) \max_{i\in\{3,5\}} f_{i,5} + p\,q(1-p)(1-q)^2 \max_{i\in\{4,5\}} f_{i,5} + p\,q(1-q)^3 f_{1,5} + \\
& p^2\,q(1-q)^2 f_{2,5} + p^2\,q(1-q)^2 f_{3,5} + p^2\,q(1-q)^2 f_{4,5} + p\,q(1-q)^3 f_{5,5}\}
\end{aligned}
$$

The original maximization problem can therefore be divided into the sub-problems of finding the constrained maxima of $F_K(K, \underline{x}^K)$, $K = 1, \ldots, K_{max}$, where $\underline{x}^K$ is the variable and K is fixed.

Still, the maximization of F_K is made difficult by the presence of the max functions. To get around this, we partitioned $\Re^K$ into 2^{K-1} sub-spaces σ_i characterized by $\{x_j < x_k < \ldots < x_l\}$ where $\{j, k, \ldots, l\}$ are all the permutations of the set $\{1, \ldots, K\}$. Considering F_K over each of these sub-spaces allowed to remove the max functions. Moreover, we could identify the subspaces in which the maxima of F_K occur. These sub-spaces depend on the values of p and q. And we could finally rewrite the optimization problem P1 into the problem P2:

$$
\underset{1 \le K \le K_{max}}{\text{maximize}} \quad \max_{\underline{x}^K \in [0, X_{max}]^K} F_K(K, \underline{x}^K)
$$

subject to the same constraints as P1. Where the functions F_K are defined as follows:

$$
\begin{aligned}
F_1 &= \tfrac{q}{p+q} f_{1,1} \\
F_2 &= \tfrac{q}{p+q} f_{1,2} + \tfrac{p\,q}{p+q} f_{2,2}
\end{aligned}
$$

K	Redundancy	$PLR_{after\,reconstr}$
1	none	$p/(p+q)$
2	-1	$p(1-q)/(p+q)$
3	-2	$(p^2q + p(1-q)^2)/(p+q)$
	-1-2	$(p(1-q)^2)/(p+q)$
4	-3	$(p(3pq - p^2q - 2q^2p + (1-q)^3))/(p+q)$
	-1-3	$(p(1-q)(pq + (1-q)^2))/(p+q)$
	-1-2-3	$p(1-q)^3/(p+q)$
5	-4	$(p^2q((1-p)^2 + 2(1-p)(1-q)$ $+3(1-q)^2) + p^3q^2 + p(1-q)^4)/(p+q)$
	-2-4	$(p^3q^2 + 2p^2q(1-q)^2 + p(1-q)^4)/(p+q)$
	-1-2-4	$(p^2q(1-q)^2 + p(1-q)^4)/(p+q)$
	-1-2-3-4	$(p(1-q)^4)/(p+q)$

TABLE I

Loss rates after reconstruction. Note: in the column Redundancy, -1-2 means that packets n-1 and n-2 were sent in packet n.

if $p + q < 1$

$$
\begin{aligned}
F_3 &= \tfrac{1}{p+q} * \{q\,f_{1,3} + p^2\,q f_{2,3} + p\,q(2-p-q) f_{3,3}\} \\
F_4 &= \tfrac{1}{p+q} * \{q\,f_{1,4} + p^2\,q(2-p-q) f_{2,4} + p^2\,q(1-q) f_{3,4} + p\,q(3- \\
&\quad 3p + p^2 + 2pq - 3q + q^2) f_{4,4}\} \\
F_5 &= \tfrac{1}{p+q} * \{q\,f_{1,5} + p^2\,q(pq + (1-q)^2) f_{2,5} + p^2\,q(2(1-p)(1-q) + \\
&\quad (1-q)^2 + (1-p)^2) f_{3,5} + p^2\,q(1-q)^2 f_{4,5} + p\,q((1-p)(1- \\
&\quad q)^2 + (1-p)^2(1-q) + (1-q)^3 + 2pq(2-p-q) + (1-p)^3) f_{5,5}\}
\end{aligned}
$$

if $p + q > 1$

$$
\begin{aligned}
F_3 &= \tfrac{1}{p+q} * \{q\,f_{1,3} + p\,q f_{2,3} + p\,q(1-q) f_{3,3}\} \\
F_4 &= \tfrac{1}{p+q} * \{q\,f_{1,4} + p\,q f_{2,4} + p\,q(1-q) f_{3,4} + p\,q(1-q)^2 f_{4,4}\} \\
F_5 &= \tfrac{1}{p+q} * \{q\,f_{1,5} + p\,q f_{2,5} + p\,q(1-q) f_{3,5} + p\,q(1-q)^2 f_{4,5} + \\
&\quad p\,q(1-q)^3 f_{5,5}\}
\end{aligned}
$$

The constraint on the packet loss rate after reconstruction can be formulated as a set of constraints on the values of $\underline{x}^K = (x_1, \ldots, x_K)$. Actually, Table 1 shows $PLR_{after\,reconstr}$ for a given amount of redundancy. Hence, a maximum value for $PLR_{after\,reconstr}$ amounts to imposing a minimum amount of redundancy. If r_0 denotes the minimum rate used to encode audio samples, $PLR_{after\,reconstr} < PLR_{max}$ is equivalent to $x_i \ge r_0$, for all i in the minimal set of copies which yield a packet loss rate smaller than PLR_{max}.

Once the formulation of the original problem is simplified, the maximization of the objective functions F_K,$K = 1, \ldots, K_{max}$ can be carried out using classical methods, the choice of method dependant on the utility function $f(x, y)$. If $f(x, y)$ is differentiable with respect to x and strictly concave, the maximizing values $\underline{x}^K$ can be found by the Lagrangian method. If $f(x, y)$ is a non-linear concave function of x, numerical methods such as SQP (Sequential Quadratic Programming) can be used. In addition, if $f(x, y)$ is a piecewise linear function of x, linear programming methods provide a solution to the maximization problem.

References

[1] J-C. Bolot, S. Fosse Parisis, and D. Towsley, “Adaptive FEC-based error control for internet telephony,” in *Infocom’99*, March 1999.

[2] M. Podolsky, C. Romer, and S. McCanne, “Simulation of fec-based error control for packet audio in the internet,” in *IEEE Infocom’98*, April 1998.

[3] J-C. Bolot and A. Vega Garcia, “Control mechanisms for packet audio in the internet,” in *IEEE Infocom’96*, March 1996.

[4] T. J. Kostas, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, and J. Mahler, "Real-time voice over packet-switched networks," *IEEE Network*, pp. 18–27, January/February 1998.

[5] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," in *Proc. SIGCOMM'99*, September 1999, pp. 109–122.

[6] Y. Moret and S. Fdida, "A proportional queue control mechanism to provide differentiated services," in *International Symposium on Computer System*, Belek, Turkey, 1998 October.

[7] P. Hurley, J.-Y. Le Boudec, and P. Thiran, "The asymmetric best-effort service," in *IEEE Globecom 1999*, Rio de Janeiro, Brazil, Dec. 1999.

[8] S. Keshav, *Congestion Control in Computer Networks*, Ph.D. thesis, UC Berkeley TR-654, September 1991.

[9] P. Hurley, M. Kara, J.-Y. Le Boudec, and P. Thiran, "Abe: Providing a low-delay service within best-effort," Tech. Rep. DSC/2001/011, EPFL-DI-ICA, February 2001.

[10] S Floyd, "TCP and explicit congestion notification," *ACM Computer Communications Review*, vol. 21 n0 5, pp. 8–23, 1994.

[11] "Network simulator," Available from http:www-mash.cs.berkeley.edu.ns.

[12] C. S. Perkins, O. Hodson, and V. Hardman, "A survey of packet-loss recovery techniques for streaming audio," September/October 1998.

[13] R. Blahut, *Theory and Practice of Error control codes*, Addison-Wesley, 1993.

[14] N. Shacham and P. Mc Kenney, "Packet recovery in high-speed networks using coding and buffer management," in *Proc. IEEE Infocom'1990*, June 1990, vol. 1, pp. 124–131.

[15] D. R. Figuereido and E. de Souza e Silva, "Efficient mechanisms for recovering voice packets in the internet," in *Proc. IEEE Globecom'99*, 1999, pp. 1830–1836.

[16] V. Hardman, A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the internet," in *Proceedings of INET'95*, June 1995.

[17] H. Sanneck, "Concealment of lost speech packets using adaptive packetization," in *Proceedings IEEE Multimedia Systems 1998*, Austin, TX, June 1998.

[18] International Telecommunications Union, "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (cs-acelp)," Tech. Rep., ITU-T Recommendation G.729, March 1996.

[19] H. Sanneck and N. Le, "Speech property-based FEC for Internet Telephony applications," in *Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference (MMCN)*, San Jose, CA, January 2000, pp. 38–51, ftp://ftp.fokus.gmd.de/pub/glone/papers/Sann0001:Speech-FEC.ps.gz.

[20] J-C. Bolot and A. Vega Garcia, "The case for FEC-based error control for packet audio in the internet," in *to appear in ACM Multimedia Systems*.

[21] V. Paxson, "End-to-end internet packet dynamics," in *Proc. ACM Sigcomm'97*, september 1997, Cannes, France.

[22] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," in *Proc. ACM Sigcomm'93*, August 1993, pp. 189–199, San Francisco, CA.

[23] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of temporal dependence in packet loss," in *Proc. IEEE Infocom'99*, March 1999, New York, NY.

[24] J. Mahdavi and S. Floyd, "TCP-Friendly unicast rate-based flow control," in *Draft posted on end2end mailing list*, Jannuary 1997, http://www.psc.edu/networking/papers/tcp_friendly.html.

[25] S. Jacobs and A. Eleftheriadis, "Providing video services over networks without quality of service guarantees," in *RTMW'96*, October 1996, Sophia Antipolis, France.

[26] I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP emulation at receivers - flow control for multimedia streaming," Technical report, Department of Computer Science, NCSU, April 2000.

[27] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proc. IEEE Infocom'99*, March 1999, New York.

[28] D. Sisalem and H. Schulzrinne, "The loss-delay adjustment algorithm: A TCP-Friendly adaptation scheme," in *NOSSDAV'98*, July 1998, Cambridge,UK.

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

[30] W. Tan and A. Zakhor, "Error resilient packet video for the internet," in *ICIP'98*, October 1998, vol. 3, pp. 458–462.

[31] J-C. Bolot and T. Turletti, "Experience with rate control mechanisms for packet video in the internet," *ACM Computer Communication Review*, vol. 28, no. 21, January 1998.

[32] M. Mathis, J. Semke, J. Madhavi, and T. Ott, "Macroscopic behavior of TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, July 1997.

[33] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. SIGCOMM'98*, 1998.

[34] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM'2000*, August 2000.

[35] T. Cover and J. Thomas, *Elements of Information theory*, John Wiley and Sons, 1991.

[36] J. Tribolet, P. Noll, B. McDermott, and R. Crochiere, "A study of complexity and quality of speech waveform coders," April 1978, pp. 586–590, Tulsa.

[37] L. Breslau and S. Shenker, "Best-effort versus reservations: A simple comparative analysis," *Proc. of ACM SIGCOMM'98*, August 1998, Vancouver, Canada.

[38] R. V. Cox and P. Kroon, "Low bit-rate speech coders for multimedia communication," *IEEE Communications Magazine*, pp. 34–41, December 1996.

[39] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proc. IEEE Infocom'94*, 1994.

[40] N. Jayant, "Effects of packet loss on waveform coded speech," in *Proc. 5th Int. Conference on Computer Communications*, October 1980, pp. 275–280.

[41] V. A. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 821–834, May 1993.

[42] D. Sisalem, F. Emmanuel, and H. Schulzrinne, "The direct adjustment algorithm: A TCP-Friendly adaptation scheme," Technical report, GMD-FOKUS, August 1997.

[43] "Robust audio tool," http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/.