G.729 Error Recovery for Internet Telephony

Jonathan D. Rosenberg
Lucent Technologies, Bell Laboratories
&
Columbia University

Abstract

This memorandum discusses the use of the ITU G.729 CS-ACELP speech coder on the Internet for telephony applications. In particular, the memo explores issues of error resiliency and recovery. In particular, the paper considers two questions. First, given N consecutive frame erasures (due to packet losses), how long does the decoder take to resynchronize its state with the encoder? What is the strength of the resulting error signal, both in terms of objective and subjective measures? The second issue explores which particular factors contribute to the strength of the error signal: the distortion of the speech due to the incorrect concealment of the erasures, or the subsequent distortion due to the loss of state synchronization, even though correct packets are being received. Both objective and subjective measures are used to characterize the importance of each of these two factors.

Table of Contents

Introduction

As Internet connectivity becomes more and more pervasive, its use for teleconferencing applications becomes increasingly attractive. At the moment, much of this interest stems from an asymmetry in pricing structures between Internet access (which is generally flat rate), and normal telephone service (which is billed per minute). Users are willing to accept the lower quality of Internet telephony because of its reduced cost. However, a number of factors are changing the motivations for this technology. Improvements in service, due to increasing bandwidths and more reliable equipment, are improving the quality of internet telephony. New protocols, such as RSVP [3] (used to reserve network resources), RTP [2] (used to transport voice and video), and PIM [1] (used for multicast), are also improving quality and interoperability. Furthermore, integrating voice and video services onto the data network allows for both cheaper infrastructure and improved features [5].

With these changing factors, it is becoming more and more important to consider how to deliver the highest quality voice on the Internet. As a transmission channel, the Internet poses a number of unique difficulties:

?? Delays can be large; furthermore, they can vary, and there are no guarantees on their behavior
?? Packets are often lost; the loss statistics can vary, and there are no guarantees on the behavior
?? Access link speeds are often limited to 28.8 kbps

Overview of ITU G.729

G.729, also known as CS-ACELP (Conjugate Structure Algebraic Code Excited Linear Prediction), is specified by the ITU (International Telecommunications Union) [4]. It compresses speech from 16 bit, 8kHz samples (128 kbps) to 8 kbps, and was designed for cellular and and networking applications. It provides "toll quality" speech (that is, as good as the telephone network), works well with background noise, and has been designed to perform well under error conditions. It, along with G.723.1 (a 5.3/6.3 dual mode speech coder), are the main contenders for the baseline codec for internet telephony.

G.729 fits into the general category of CELP (Code Excited Linear Prediction) speech coders [6]. These coders are all based on a model of the human vocal system. In that model, the throat and mouth are modeled as a linear filter, and voice is generated by a periodic vibration of air exciting this filter. In the frequency domain, this implies that speech looks somewhat like a smooth response (called the envelope), modulated by a set of discrete frequency components. CELP coders all vary in the manner in which the *excitation* is specified, and the way in which the coefficients of the filter are represented. All of them generally break speech up into units called *frames*, which can be anywhere from 1ms to 100ms in duration. For each frame of speech, a set of parameters for the model are generated and sent to the decoder. This implies that the frame time represents a lower bound on the system delay; the encoder must wait for at least a frames worth of speech before it can even begin the encode process. In G.729, each frame is 10ms, or 80 samples, in duration. This frame is further broken into two 5ms subframes. The filter parameters are specified just once for each frame, but each subframe has its own excitation specified. It is also important to note that speech can generally be classified into two types: voiced and unvoiced. Voiced sounds, such as b,d, and g, are generated from the throat, where as unvoiced sounds, such as th, f, and sh, are generated from the mouth. The model works better for voiced sounds, but the excitation can be tailored for voiced or unvoiced so that it works in both cases.

The approach for finding the filter parameters and excitation is called *analysis by synthesis*. The encoder searches through the parameter space, performing the decode operation in each loop of the search. The output of the decoder (the synthesized signal), is compared with the original speech signal. The parameters which yield the closest match are then chosen, and sent to the decoder. In this fashion, we have analyzed the signal by repeatedly synthesizing the output of the decoder, and thus the name analysis by synthesis.

First, we discuss how G.729 computes and transmits the filter coefficients. Then we discuss the excitation.

Filter Coefficients

In G.729, the filter is a 10th order all-pole filter. Since it is used to synthesize voice, it is also called the *synthesis filter*. Its inverse, the *analysis filter*, is an all-zero FIR filter (which we done by A(z)). When speech is passed through it, the result is the excitation for that speech. In fact, the analysis filter can be thought of as a linear predictor, the output of which is the error signal in predicting the speech from the past 10 samples. With this in mind, the problem of finding the coefficients of the analysis filter reduces to finding the optimal 10th order linear predictor for a given signal. This problem is well known, and the solution is a function of the correlation matrix of the speech.

The correlation function, however, is likely to vary over time. In each frame, it is re-measured over a 30ms interval. This interval consists of 15ms from the past, 10ms from the current frame, and 5ms from the future. Of course, this *look-ahead* of 5ms requires the encoder to wait an additional 5ms beyond the 10ms frame delay. This means that the total encoding delay, also known as the *algorithmic delay*, is 15ms. Instead of just computing the correlation directly from those speech samples, a window, called the LP-analysis window, is applied to the samples. The window is half of a Hamming window on one side, and a quarter cosine cycle on the other side. The curved shape of the window helps emphasize the current time as opposed to the future or past when computing the correlation function.

One further step is taken before the correlation coefficients are used to generate the filter coefficients. For high pitch speech signals (such as those from females), the modulation frequency of the spectral envelope is higher than for than for lower pitch signals. Thus, the LP analysis will tend to result in filters which underestimate the envelope at frequencies between the pitch periods. To resolve this, the correlation coefficients are multiplied by a gaussian function. This is equivalent to convolution in frequency of the spectral envelope by a Gaussian. The result is a widening of the peaks of the spectral envelope, filling in the gaps.

With the correlation function r(k) computed, the 10 LP analysis filter coefficients can be computed. The optimal coefficients a_i, i=1..10, are the solution to:

$$\sum_{i=1}^{10} a_i r(|i-k|) = -r_k$$

This well known Yule-Walker equation is readily solved with the Levinson-Durbin algorithm, which defines an iterative approach to its solution.

The next step is to quantize the filter coefficients. However, just quantizing them directly has several drawbacks. First, it is possible that the quantization noise may move one of the poles of the synthesis filter outside of the unit-circle, yielding an unstable filter. Secondly, since human perception of noise is based on frequency components, it is hard to relate the quantization noise of the coefficients to the noise that will actually be perceived. To resolve this, the coefficients are transformed into Line Spectral Frequencies, or LSF's. This is done by defining two new polynomials:

$$F_1(z) = A(z) + z^{-11}A(z^{-1})$$
$$F_2(z) = A(z) - z^{-11}A(z^{-1})$$

The LSF's are defined as the zeroes of these polynomials. These two polynomials have several important characteristics:

1. Their zeroes lie on the unit circle
2. Their zeroes alternate each other
3. For any two polynomials defined as above, with their zeroes on the unit circle and alternating, the filter A(z) is minimum phase, and therefore, its inverse, the synthesis filter, is stable.
4. A change in any LSF causes a change in the shape of the analysis filter only in a small frequency range around the frequency of that LSF.

Because of property 3, the decoder can easily verify stability of the filter by making sure the zeroes are on the unit circle and alternate. Property 4 allows the quantization of the LSF to relate to the frequency response of the synthesis filter.

To reduce the bandwidth, the encoder and decoder predict the values of the LSF's via a 4th order moving average. Two predictors are possible; the encoder chooses which one to use and indicates it with a bit in the bitstream. After prediction, the prediction error is computed. This error is sent to the decoder by vector quantizing it. The vector quantization proceeds in two stages. In the first, a 10-dimensional codebook (recall their are 10 coefficients) containing 128 entries is searched, and the "best" one is chosen. "Best" is defined here as the entry which results in the minimum mean square error between the correct LSF's and their quantized versions. This 10-dimensional vector is then subtracted from the original LSF's. The resulting 10 dimensional difference is split into two 5 dimensional vectors. The best match for the first vector is found (best here is defined as minimizing a weighted m.s.e) from a second codebook, and the best match for the second vector is found from a third codebook. This second codebook is 5-dimensional, and contains 32 entries, as does the third. This two stage structure is called a conjugate structure, and represents the CS in the codec's name. Note that 7+5+5=18 bits are needed for the vector quantization, and another bit to specify which moving average function is used.

In the decoder, the LSF's are received and decoded. However, for the first subframe in the frame, the LSF's are interpolated as the average of the LSF's for the current and previous frames. The second subframe uses the LSF's received for the current frame.

Excitation

The next step is to compute the excitation. This is done separately for each subframe. In each, the excitation is represented as the sum of two components. The first is a delayed version of the excitation used so far, and the second is a signal with four impulses at various positions. The first component is called the *adaptive codebook* contribution, and it models the periodicity in the speech. Therefore, this delay is actually the pitch delay in the speech signal.

The first step in the process is to compute the pitch delay. This is done by computing the autocorrelation of the speech (weighted to emphasize various frequency characteristics), and finding the least maximum. Searching for the least maximum ensures that multiples of the pitch delay are not used. This is called an open-loop pitch analysis. With this quantity, a search is done in a region around the open loop pitch delay to find the best pitch. Best is defined by filtering the previous excitation (delayed by the appropriate amount) through the LP synthesis filter. The result is correlated with the actual speech signal, and divided by the magnitude of the output of the synthesis filter (thus the gain is eliminated from the search). The delay which maximizes this quantity is chosen. The gain is then computed directly for the optimal excitation. The output of the synthesis filter using the optimally delayed and amplified excitation is then subtracted from the desired speech signal, and the difference, called the target, is then used to find the second part of the excitation.

The second part of the excitation is referred to as the fixed codebook contribution. The excitation consists of four impulses. Each impulse has an amplitude of either plus or minus one, and can sit at a fixed set of positions (the set of positions is different for each impulse). These pulses are then filtered through a simple harmonic filter. A search is done, first identifying the ideal amplitudes (plus or minus one), and then the positions. As before, the search is executed by filtering the excitation through the synthesis filter, and computing the product of the result with the target. This is then divided by the energy in the output of the synthesis filter (again, eliminating the gain from the search), resulting in the search metric. The set of amplitudes and positions which maximize this metric are chosen. Finally, the gain is computed directly.

For each subframe, a number of parameters have now been computed: the pitch delay, the adaptive codebook gain, the fixed codebook excitation (consisting of impulse positions and signs), and the fixed codebook gain. These parameters are then quantized and sent to the decoder. The pitch delay is directly represented with 8 bits in the first subframe. In the second subframe, the pitch delay is sent as the difference from the pitch delay in the first subframe. This requires 5 bits. The fixed codebook contribution is also sent directly, using 4 bits for the signs and 13 bits for the positions. What remains are the gains. The fixed codebook gain is predicted from previous frames, and a multiplicative gain factor to compensate for the prediction error is transmitted.

The gain factor and fixed codebook gain are jointly vector quantized using a two stage vector quantization process. The first stage consists of a 3 bit two dimensional codebook, and the second stage consists of a 4 bit two dimensional codebook. The sum of the two codewords is used to represent the gain factor and fixed codebook gain.

Post Processing

Once the decoder receives and reconstructs the speech signal, it applies post processing to clean it up. The post processing consists of four components:

1. A long term postfilter, denoted $H_p(z)$
2. A short term postfilter, denoted $H_f(z)$
3. A Tilt compensation filter, denoted $H_t(z)$
4. A gain compensation factor, denoted g

The long term postfilter is constructed from the decoded gain and pitch delay parameters. Its basic function is to emphasize the speech signal in frequency bands around multiples of the pitch period. The filter is therefore constructed as a 1st order all-zero filter, with its peaks precisely at multiples of the pitch period.

The short term postfilter is designed to emphasize the formants, which are frequency bands of energy present in the synthesis filter. The postfilter is therefore derived from the synthesis filter, but with its peaks expanded to make them more predominant.

In a speech signal, tilt is defined as the general slope of the energy of the frequency domain. The tilt compensation filter attempts to adjust for distortions in this quantity caused by the short term postfilter.

Finally, the gain compensation factor is just the quotient of the energy in the un-filtered decoder output divided by the energy in the postfiltered energy. It restores the original signal strength to the speech.

Bitstream

The following table lists all of the bits which are placed in the bitstream:

Parameter Name	Number of Bits
Switched MA Predictor of LSF	1
First stage LSF VQ	7
Second stage VQ, first half	5
Second stage VQ, second half	5
Pitch Delay, First Subframe	8
Parity bit for pitch delay	1
Fixed codebook for First Subframe	13
Signs of fixed codebook for First Subframe	4
Gain codebook, stage 1, for First Subframe	3
Gain codebook, stage 2, for First Subframe	4
Pitch Delay, Second Subframe	5
Fixed codebook for Second Subframe	13
Signs of fixed codebook for Second Subframe	4
Gain codebook, stage 1, for Second Subframe	3
Gain codebook, stage 2, for Second Subframe	4
Total	**80**

Annex B

G.729 has an optional annex, Annex B [7], which specifies the use of silence suppression and comfort noise generation. In typical speech, only one person talks at a time. Therefore, speech consists of periods of talking (called talkspurts), followed by periods of silence. Additional compression can be achieved by discovering the silence periods. Older approaches would send either nothing for the silence periods, or would send a simple energy value, which the decoder would use to insert white noise. However, in environments with loud and nonstationary background noise, both approaches are inadequate.

The algorithm operates by first making a Voice Activity Detection (VAD) decision in each frame. The decision is made by keeping a running average of four quantities:

1. The LSF's during silence periods
2. The full band energy in the speech signal (computed as the logarithm of the first autocorrelation coefficient) during silence periods.
3. The low band energy in the speech signal (computed by filtering the autocorrelation coefficients), during silence periods.
4. The rate of zero crossing of the signal, during silence periods.

In each frame, the above parameters are extracted, and compared with the running averages. Depending on the magnitudes of the differences for the various parameters, an activity decision is made. Furthermore, the running averages are updated if the parameters in the current frame are less than the running averages.

The decision itself (speech or silence), is filtered, using the past two frames parameters and decisions as inputs. This ensures that sufficient hangover (i.e., speech transmission just after the end of a talkspurt) is present.

If the decision for the current frame is silence, the next step is to decide whether to send a Silence Insertion Description frame (SID), or to send nothing (a null frame). The SID frames contain a small amount of information which allow the decoder to generate comfort noise. They consist of an excitation energy (5 bits), and the prediction error for the LSF coefficients, as in G.729 (10 bits). The SID frames need only be sent when the parameters of the background noise have changed since last transmitted. The decision is made by the encoder in any way it likes, generally by comparing filter coefficient and energy changes to some thresholds.

Note that the bitstream does not contain any information about which of the three frame types are present (speech, SID, or null). This information must either be sent out of band, or can be extracted from the size of the frame (80, 15, or 0 bits, respectively).

Loss Concealment

Since G.729 was developed for environments such as cellular and data networks, an algorithm has been specified for concealing the loss of a frame. A frame is lost when the network layer indicates sufficient bit errors in the frame, or when the frame never arrives at all (due to a packet loss in the Internet, for example). When this happens, all of the parameters in the packet are interpolated from parameters from the previous frame. In particular:

1. The LSF parameters for the current frame are repeated from the previous frame.
2. The adaptive and fixed codebook gains are taken from the previous frame, but are attenuated to gradually reduce their impact.
3. The excitation depends on the classification of the previous frame as voiced or unvoiced. If the previous frame was voiced, the fixed codebook contribution is set to zero, and the pitch delay is taken as the same as the previous frame. If the previous frame was unvoiced, the adaptive codebook contribution is set to zero, and the fixed codebook contributions are selected randomly.

The effect of this interpolation will be to introduce errors into the decoded speech signal, both for the frames which are erased, and for subsequent correctly received frames, due to the divergence of encoder and decoder state. Unfortunately, quite a bit of state is maintained in the decoder, including:

1. The 4th order MAR predictor filter memories for the LSF's.
2. The past excitation signal
3. The fixed codebook energies for the past four frames, which are used to predict the fixed codebook gain
4. The adaptive codebook gain from the previous frame, which is used to generate the harmonic filter used on the fixed codebook excitation.
5. The synthesis filter memories (10th order)

The remainder of this memo discusses measurements relating to these errors.

Overview of Error Issues

As mentioned in the introduction, the Internet will introduce delays and losses into a speech stream sent over the Internet [8][9]. These losses can be bursty, which will cause consecutive numbers of codec frames to be lost. There are a number of approaches which can be taken to deal with these losses:

1. Concealment. First, the concealment strategies already in place in the decoder can be used by themselves. In this way, the decoder can deal with packet loss to some degree without requiring any extra work from either the network or the transmitter.
2. Retransmission. The decoder can inform the encoder of errors, and the encoder can retransmit the lost frames. This approach can yield perfect error performance, but with slightly higher bandwidth requirements, and substantially larger delays.
3. Resynchronization. The decoder can inform the encoder of errors, but instead of resending the lost frame, the encoder reinitializes its state, and informs the decoder about the re-initialization in the next packet. This approach does not help restore state for lost frames any more than simple concealment. However, it helps restrict the amount of time for which the encoder and decoder states will be different. The utility of this approach depends on the difference between round trip times in the network and the amount of time it takes for encoder and decoder to resynchronize unaided. If the natural resynchronization time is less than the network roundtrip time, this approach is useless. If it is substantially more, it is very helpful.
4. Forward Error Correction. This approach is really a class of algorithms. The idea is to send extra information at the encoder which can assist the decoder in recovering lost frames. A number of flavors of FEC have been explored:
    1. Redundant Encodings. It was proposed in [10] and [12]to transmit a very low bit rate coded version of the speech in packet n, in packet n+1. In general, a packet can contain many time-shifted coded versions of the speech signal. This approach helps restore the speech for a lost frame (one

    can just pick the speech out of any of the time shifted versions that have already arrived), but the state for ALL of the coded streams is distorted when a packet is lost. This approach also adds computational complexity, since a number of parallel encoders and decoders must be run. It also introduces delay, since you must wait long enough to receive the longest time delayed version of the speech.

    2. Channel Codes. Standard channel codes, such as CRC, Reed-Solomon, or parity codes can be used [11][13]. Instead of being done within a packet, they are done across multiple packets. For example, consider N packets of speech. An N+1th packet is generated by computing, in each bit position, the parity across all other packets. At the receiver, if any N-1 of the N data packets, and the parity packet, arrive, the missing packet can be reconstructed. This approach perfectly restores lost speech, and thus avoids the decoder loss of sync. However, it adds delay and requires extra bandwidth.
    3. Diversity. A variation on the redundant encodings approach is to send the same speech signal over multiple paths in the Internet. This way, if a packet is lost on one path, it may arrive on another. This approach is very bandwidth inefficient. It also requires the user to be able to select alternate paths for a packet to the same destination, which can be either impossible in some cases, or complex in others.
5. Interleaving. The data in N consecutive frames can be mixed together before transmission. This way, loss of a packet destroys only a few bits from each frame. Assuming the coder is more robust to bit errors than frame erasures (which is generally true), this approach may lessen the effect of loss. However, it does so at the expense of the substantial delays.

Of course, all of the approaches can be combined to yield any number of a variety of hybrids. Each can also be made adaptive, varying the amount of delay, redundancy, etc. as network losses vary. In general, each of the above approaches are a tradeoff among a number of different variables, which include (a) bandwidth (b) delay (c) computational complexity (d) speech quality. Which one is best is dependent on the importance placed on these four factors.

Our goal here is to answer some questions about the relative merits of these approaches as they relate to the speech quality and loss behavior of G.729. In particular, the following questions seem to be of key importance:

1. Resynchronization works only when network delays are less than the natural resynchronization time of the codec. What is this natural resynchronization time? Does it depend on the number of consecutively lost frames? How much does it vary? How can it be measured? Is it speech content dependent?
2. FEC only makes sense of the coder cannot recover via concealment. How well does this concealment work? How does it vary with the number of consecutively lost packets?

3. Use of redundant encodings restores the speech segment lost when the frames were erased, but does not correct the loss of synchronization between encoder and decoder. Which is more important, repairing this speech, and ignoring the state, or restoring the state and letting the decoder conceal the missing speech?

The next three sections discuss these three issues precisely.

Resynchronization Time

The first set of experiments was to measure the convergence time of the codec after N consecutively lost frames. The most significant difficulty in doing this is to define what is meant by convergence time. A number of possible definitions exist:

1. Compute the difference between the decoded bitstream with no frame erasures, and the decoded bitstream with erasures. After an erasure occurs, compute the amount of time until the energy in the difference falls below some threshold. The threshold can either be static or adaptive to the strength of the error signal. This approach is the simplest, requiring no interaction with the encoder or decoder except through the decoder output. It likely defines an upper bound on the "real" convergence time, whatever that may be.
2. The same approach as 1, but instead of a plain difference signal, use a perceptually weighted error signal. G.729 defines a perceptual weighting filter which is derived from the LP synthesis filter, and this can be used directly. This approach requires interaction with the encoder (or decoder) to obtain the LP filter parameters. However, it yields a realistic measure on the human perception of the convergence time.
3. Instead of computing the error between the decoded speech, the decoder state can be consider as a multidimensional vector, and a distance measure can be defined (perhaps perceptually) to compute the difference between the encoder and decoder state over time. This approach has the advantage of actually looking at the state, and not the output of that state (i.e., the decoded speech). However, it requires interaction between encoder and decoder, and it necessitates the definition of a complex distance metric.
4. Either option 1 or 2, but with the postfilter turned off. Since the postfilter does not represent decoder state (at least, state that must be synchronized with the encoder), it may get in the way of measurements based on the speech itself.

The approach used here is (1), above. It is the simplest, and it gives an upper bound on the real convergence time. In particular, the threshold was made adaptive. A simple, one-pass algorithm was used to compute the convergence time. The mse in each frame (again, the mse is between the decoded version of the unerrored speech bitstream and the decoded version of the errored bitstream. This eliminates coding loss from the computation) is computed. The starting time of the convergence period is defined as the first good frame received after a burst of erased frames. As subsequent frames are received, the maximum mse so far is noted, and the threshold is set at 1% of this quantity. If the mse in the current frame is above the threshold, the stopping time is set to the next frame. The process continues until the next burst of frame erasures. This way, the stopping time represents the last time the mse fell below the threshold. The convergence time is computed as the difference between the stopping and starting times. Since the mse was computed on a frame by frame basis, the convergence times are measured in units of frames.

The specifics of the experiment are depicted in Figure 1. The process starts out with an original speech file. The G.729 encoder is then run on the speech to generate the bitstream, with no errors. A program called **erase** was written, which reads in a bitstream, a burstsize, and a pause interval. It then erases burstsize consecutive frames, waits pause consecutive frames, erases burstsize consecutive frames, etc. In all cases, pause was set to 50 frames (1/2 of a second), which was an adequate amount of time for the decoder to resynchronize before the next loss. Erase generates a bitstream (with the erasures) as an output. It also generates a marker file. This file contains an entry for each frame in the bitstream, and it indicates whether this frame was erased or not. The marker file is used to determine when to start computing the convergence times.

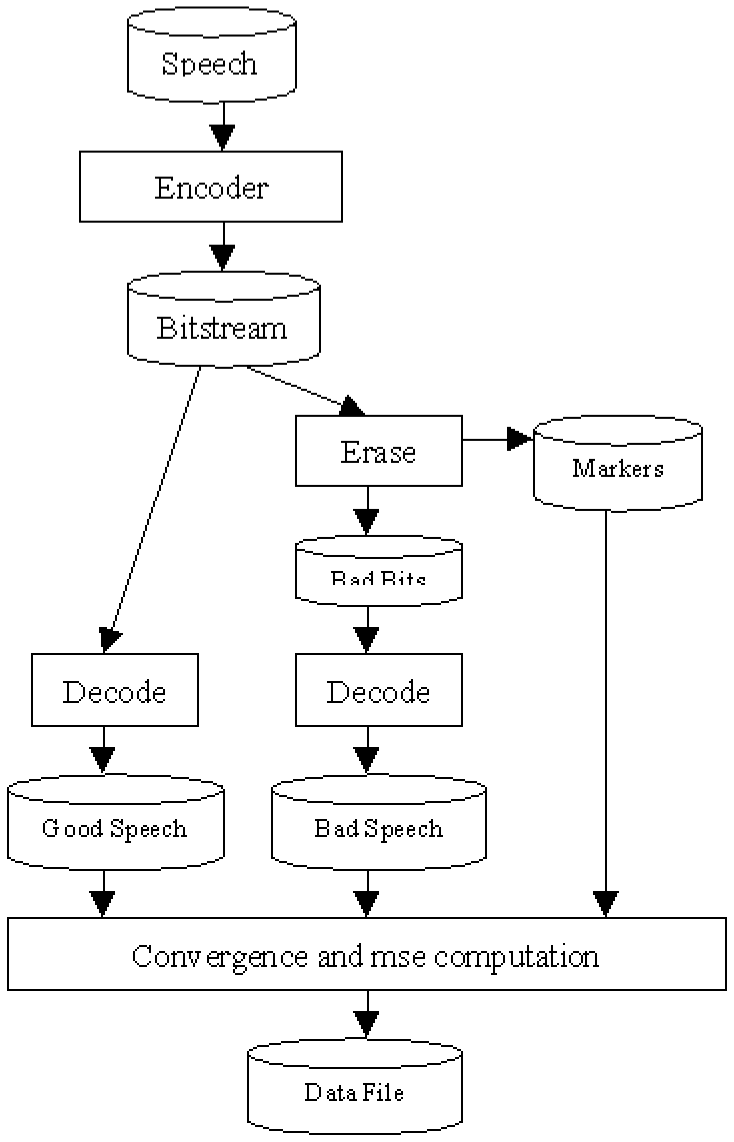

Figure 1: Convergence measurement methodology

The decoder was then run twice; once on the uncorrupted bitstream, and once on the corrupted bitstream. Another program, called mse, was written. This program takes two speech files and a marker file. Using the algorithm above, it computes the convergence times. It then outputs two files, one containing the convergence times for the erasures in the frame, and another containing mse information (discussed in the next session).

To execute the above experiment, a simple tcl script was written, called runtest.tcl. The script runs the above programs on all of the bitstreams it finds in a specific directory. It takes as an input the burstsize to use. All of the experiments were conducted using some of the speech files in the ITU corpus. In particular, the speech segments f15.d, f17.d, f26.d, f34.d, f43.d, f44.d, m19.d, m27.d, m31.d, m33.d, m41.d, and m51.d were used. The files beginning with f are female speakers, and the ones beginning with m are male speakers. Each speech segment is approximately 2 to 8 seconds in duration, with varying content and background noise. Each of these files contain 16 bit signed linear PCM speech samples at 8khz. Another script, runcode.tcl, was written to run the G.729 encoder on all of these files to generate the bitstreams needed by runtest.tcl.

After runtest.tcl executes the experiment on all of the speech files, it concatenates the convergence time files into a single file. It then executes another program called makehist2. This program takes a file of convergence times, and generates a histogram from them. The histogram quantizes the convergence times into the smallest regions possible, given that 50 regions were allowed to cover the entire range of convergence times.

The script was executed for burst sizes of one through five, inclusive. The resulting histograms are charted in Figure 2 and Figure 3. The first plots the frequency of convergence times vs. number of frames for 1 consecutive frame erasure, and for 5 consecutive frame erasures. The second figure shows the same, but for 2,3 and 4 consecutive frame erasures. Note that there appears to be no significant change in the convergence times as the burst size varies.

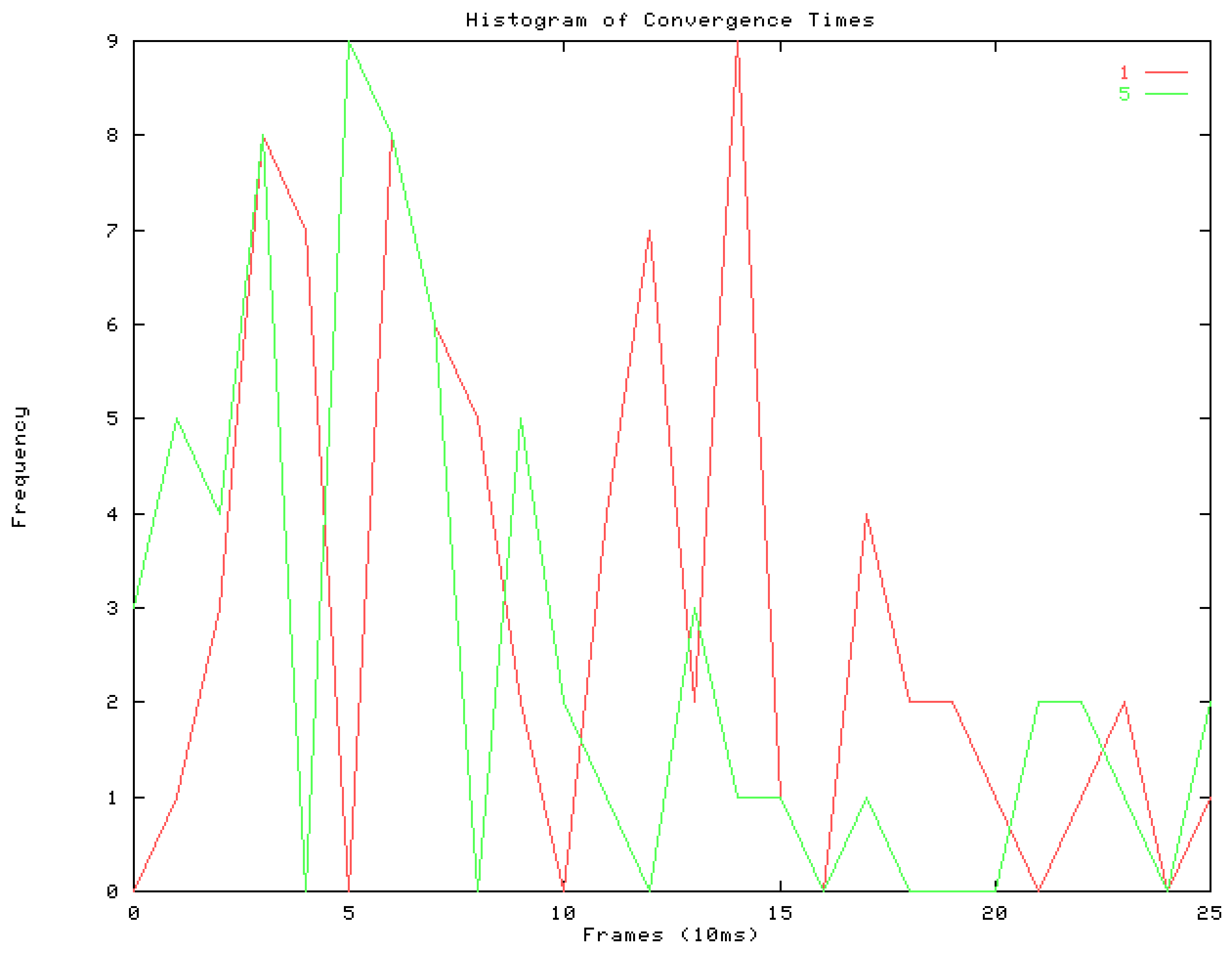

Figure 2: Histogram of convergence times for bursts of 1 and 5 frame erasures

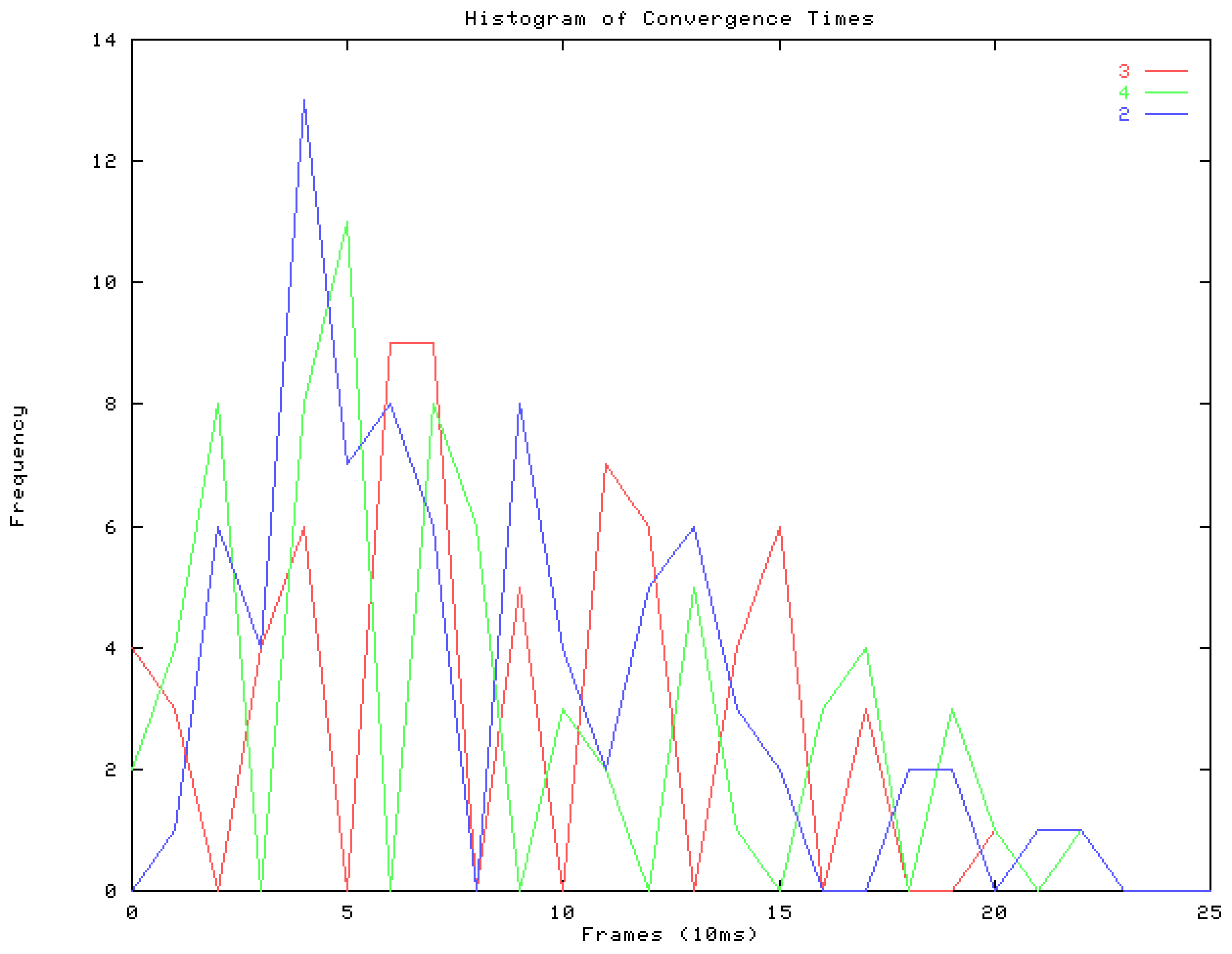

Figure 3: Histogram of convergence times for bursts of 2,3 and 4 frame erasures

This fact is quantitatively demonstrated in Table 1, which shows the average and standard deviation of the number of frames until convergence, as a function of burst size.

Burst Size	Mean Time until Convergence (in frames)	Standard Deviation
1	10.26	7.92
2	8.21	6.51
3	8.22	7.39
4	7.27	6.93
5	9.25	9.39

Table 1: Mean and S.D. of convergence times

The wild variation in the convergence times attests to the fact that they are extremely speech dependent. However, given that frame sizes are 10ms, the average convergence time is between 70 and 100ms. When comparing this with network delays, which range from 10ms to possibly several seconds on the Internet, it would seem that the resynchronization approach is not the best idea, particularly in wide area connections, where the delays are generally over 100ms.

Concealment Performance

The next issue addresses how well the concealment algorithm in G.729 performs. If it works extremely well, the need for FEC and other recovery mechanisms (including retransmission) may be alleviated. As in the previous section, measuring the performance of the mechanism is no small task. Any of a number of objective measures (some of which have been mentioned above) and subjective measures are possible. In this study, the performance of the concealment algorithm is measured in two ways:

1. The average mse over the convergence time (as defined above)
2. A subjective evaluation using MOS (mean opinion score)

Computing the average mse was a straightforward extension to the code and methods discussed above. The program mse simply adds up the mse in each frame as it processes the speech files. Any frame whose mse is above the threshold (as defined, 1% of the maximum mse seen so far), is counted in the average. The resulting average mse is written to a file, where the makehist program converts them to a histogram.

Figure 4 and Figure 5 plot the density of the mse in the error signal. Figure 4 depicts the energy density for burst sizes of 1 and 5 frame erasures, and Figure 5 depicts the energy density for burst sizes of 2, 3 and 4 frame erasures. Note that in this case, there is a noticeable increase in the error energies with increasing burst sizes.

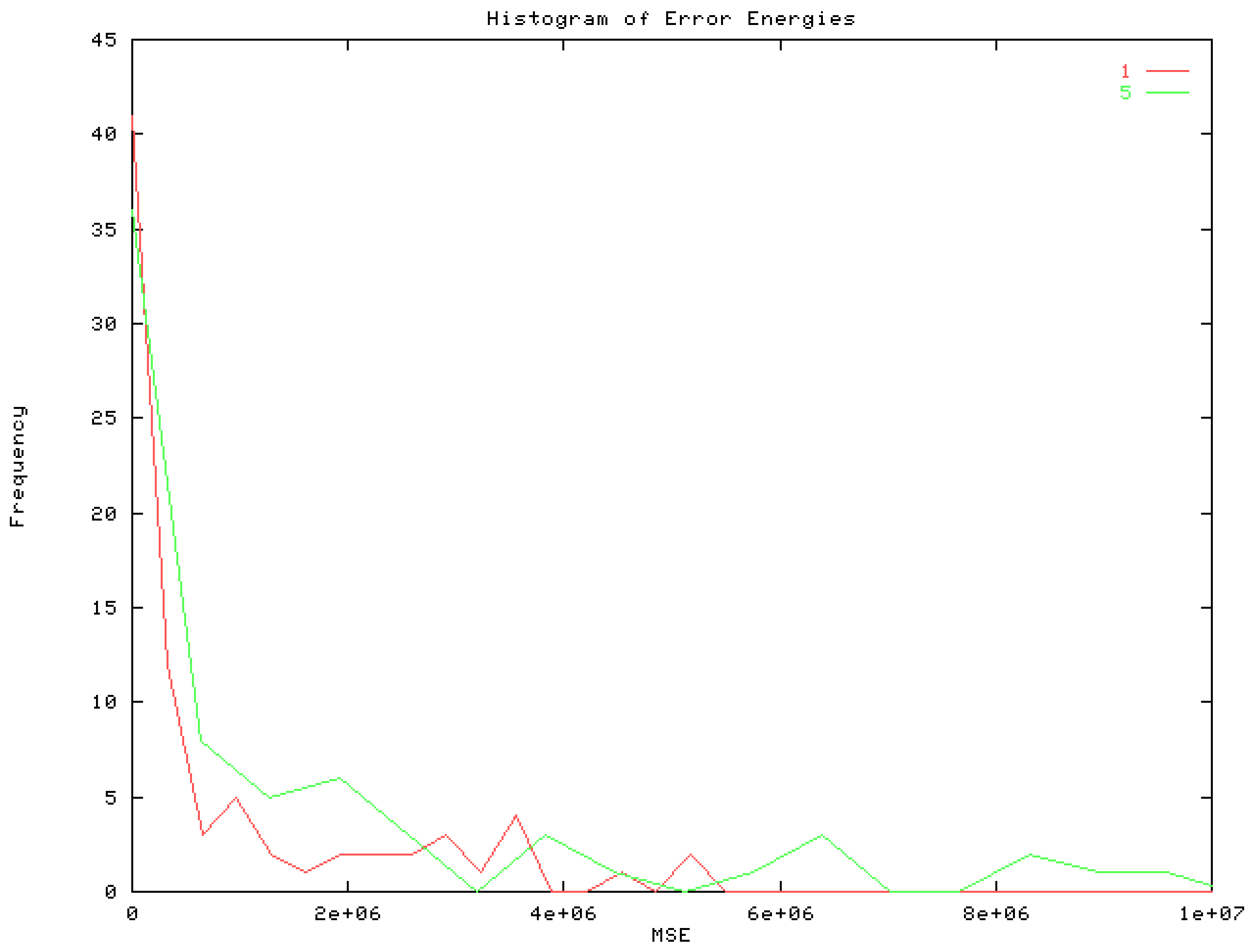

Figure 4: Density of the MSE for burst sizes of 1 and 5

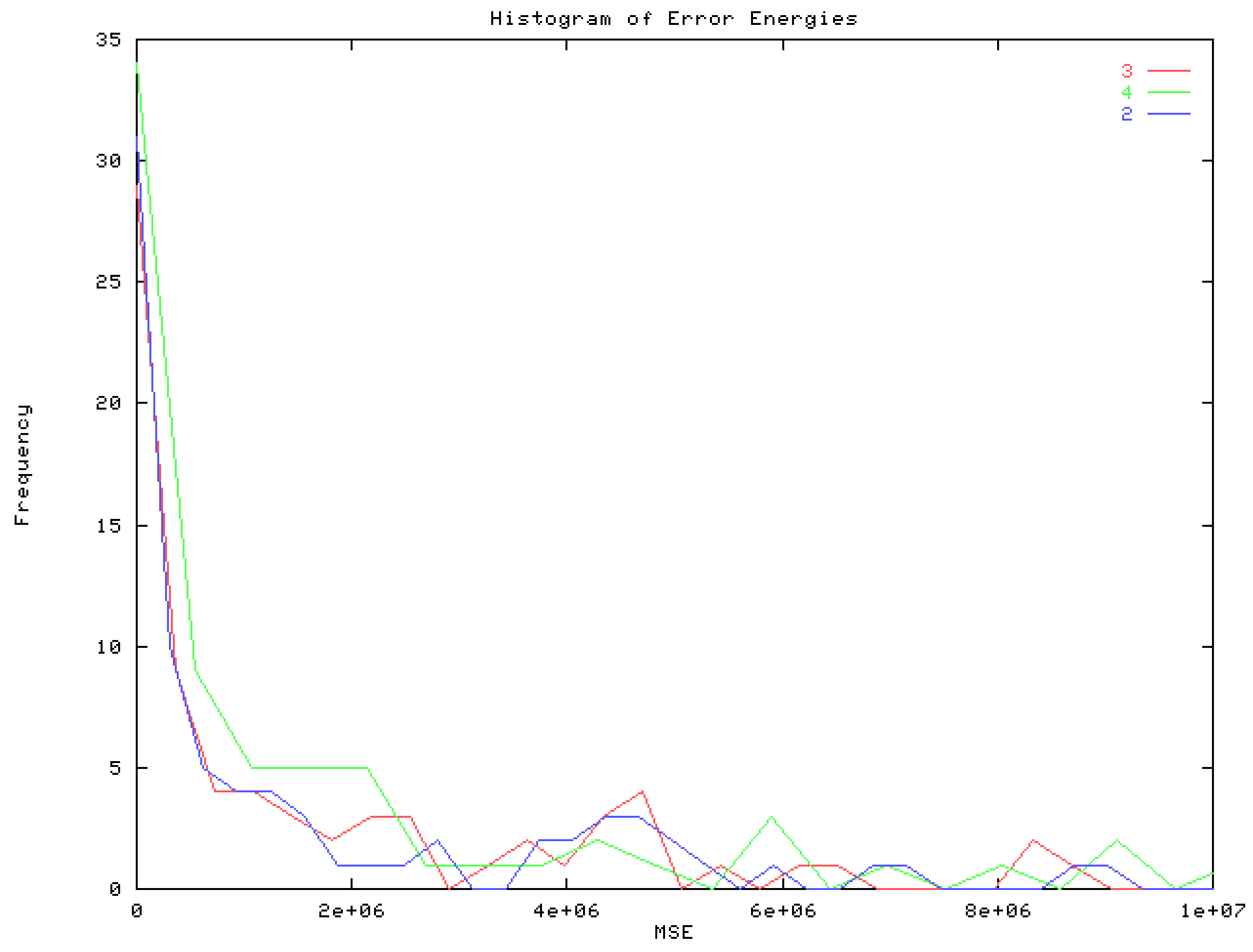

Figure 5: Density of the MSE for burst sizes of 2, 3, and 4

These observations are verified by the computation of the average and standard deviation of the energy in the error signal for the five different burst sizes. The results are depicted in Table 2.

Burst Size	Mean Energy in Error	S.D of Energy in Error
1	1.4×10^6	2.9×10^6
2	2.3×10^6	3.6×10^6
3	2.6×10^6	4.1×10^6
4	2.6×10^6	4.5×10^6
5	2.7×10^6	5.0×10^6

Table 2: Mean and S.D of MSE

The results indicate a sharp jump in the error energy when the burst size increases from 1 to 2. Subsequent increases in the burst sizes cause further increases in the MSE, but by less severe amounts. The variability in the error also appears to increase with increasing burst size. The conclusion would appear to be that the concealment strategy works well for a single erased frame, but performs less well as the number of consecutively lost frames increases.

These numerical results were backed up by subjective testing. To perform the test, 12 non-experts listened to a set of speech samples. The speech samples consisted of six versions of the same speech: the decoded speech with no errors in the bitstream, and the decoded speech with instances of 1, 2, 3, 4, and 5 consecutive frame erasures. Using the error-free decoded speech as an ideal point, the subjects were asked to rate the five other samples on a scale of 1 to 5, with 5 being as good as the error free decoded signal, and 1 being unacceptable. The test was run using the sequence f15.d, which is a female voice speaking a sentence roughly seven seconds in duration. The public domain tool sox was used to convert the 16 bit linear PCM decoder output to the .wav format used in Windows PC's. The subject evaluated the speech independently, without being told the nature of the different samples. Each subject wore headphones to eliminate room noise. The results of the analysis are depicted in Table 3.

Burst	1	2	3	4	5
Mean MOS Score	4.208333	3.275	2.425	2.083333	1.7
S.D in MOS	0.660124	0.916629	1.198697	1.426437	1.585875

Table 3: Subjective Evaluation of quality of decoded speech vs. consecutive frame erasures

The table shows results strikingly similar to the objective measures. With a single frame erasure, the speech quality is slightly worse, but still very good. However, there is a sharp drop in the speech quality as soon as two consecutive frame erasures occur. Subsequent increases in the burst size reduce the speech quality more, but by decreasing amounts.

You can click here to take the subjective test yourself.

The conclusion here is that concealment works well for a single frame erasure, but not well enough for multiple consecutive frame erasures. Since losses in the Internet often occur in bursts, additional error recovery techniques may be desirable.

The results of this section and the previous one can also be seen readily by looking at the actual error signal. Figure 6 depicts the mse from time 45 to 75 for the speech segment f15.d when 1 and 5 consecutive frame erasures occur. As the sample path indicates, the average energy in the error signal is greater for the 5 consecutive loss case, but both have roughly equivalent convergence times.

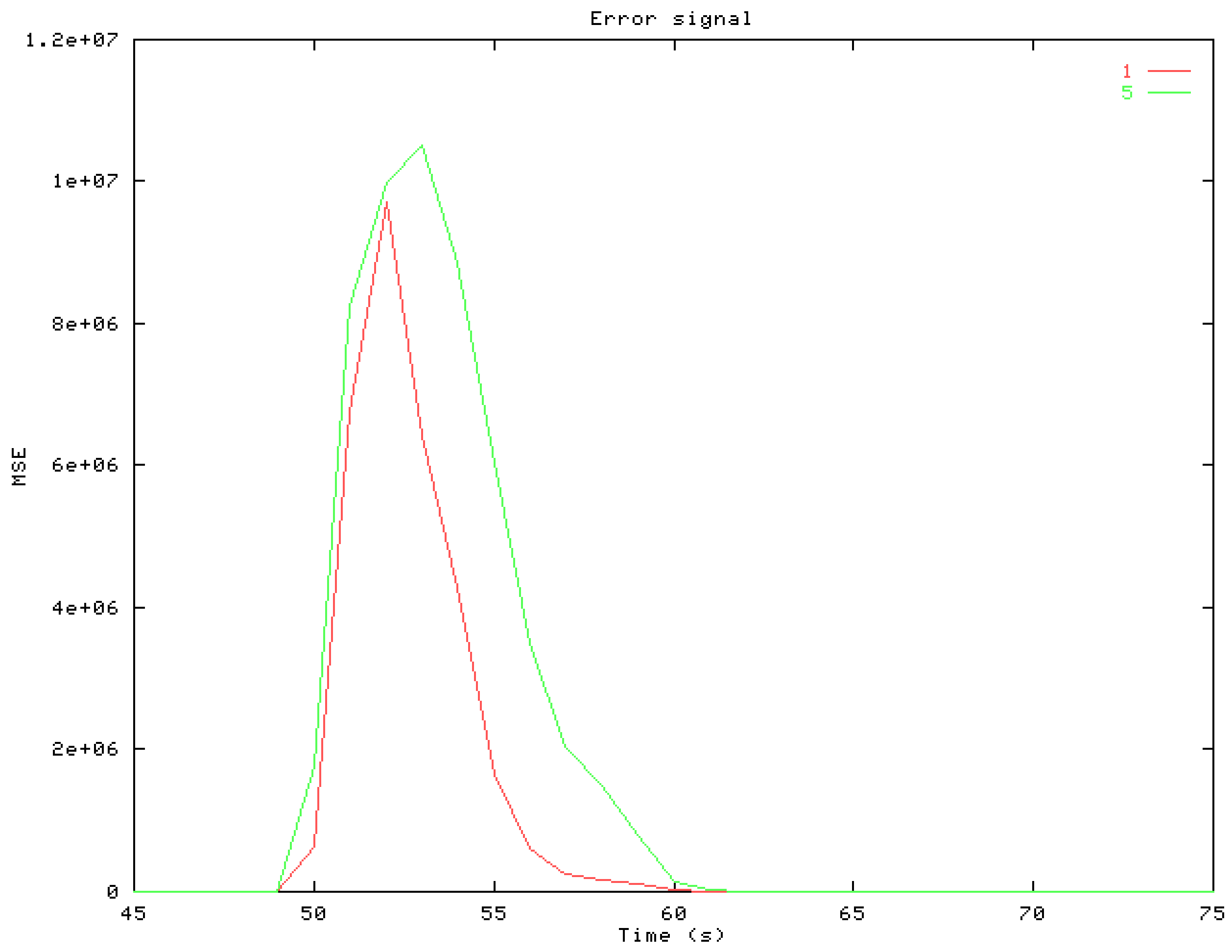

Figure 6: Comparison of error signal sample paths for 1 and 5 consecutive frame erasures

State vs. Speech

The final issue consider here is whether it is more important to recover from frame erasures by reconstructing the actual lost speech, ignoring the state of the decoder (as is done in [10]), or whether it is more valuable to restore the decoder state after a frame erasure without worrying about the quality of the speech generated for the erased frames.

To make such a determination, two speech signals were generated. The first one is the output of the decoder with no frame erasures. The second is the output of the decoder with instances of consecutively erased frames. Lets say frames N and N+1 only were erased. Two new speech signals, called mix1 and mix2, are then generated. The signal mix1 is formed by replacing frames N and N+1 in the error-free decoder output with frames N and N+1 from the errored decoder output. The signal mix2 is formed by replacing frames N and N+1 in the errored output signal with the correct N and N+1 frames from the error-free decoder output. In this fashion, mix1 emulates the decoder operation during the erased frames, but as soon as the next good frame arrives, its output is perfect again, as it would be if its state were restored. On the other hand, mix2 has correct speech output during the erased frames, as if the speech were replaced by the correct version. However, the state is not restored, since the next good frame will be generated from state updated from the concealment algorithm. The nature of these two speech signals is depicted graphically in Figure 7.

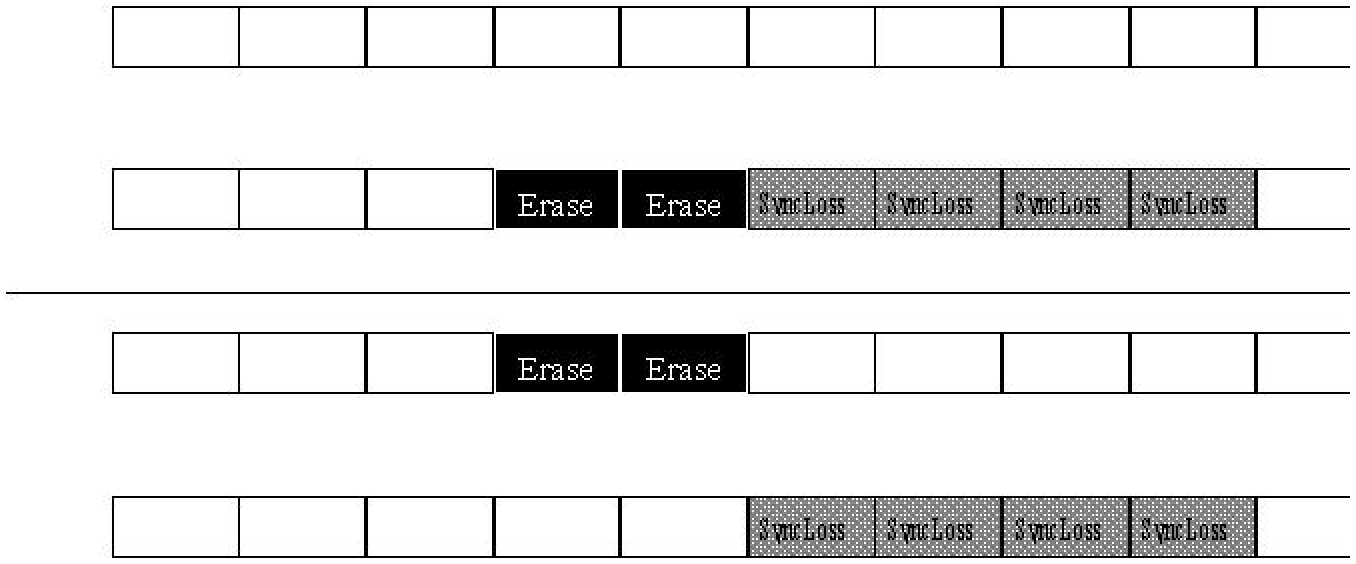

Figure 7: Mix1 and Mix2

As the figure shows, **mix1 represents recovery by state restoration, and mix2 represents recovery by speech correction.**

To actually generate and compare these two, a system similar to the one in Figure 1 was used, and it is depicted in Figure 8.

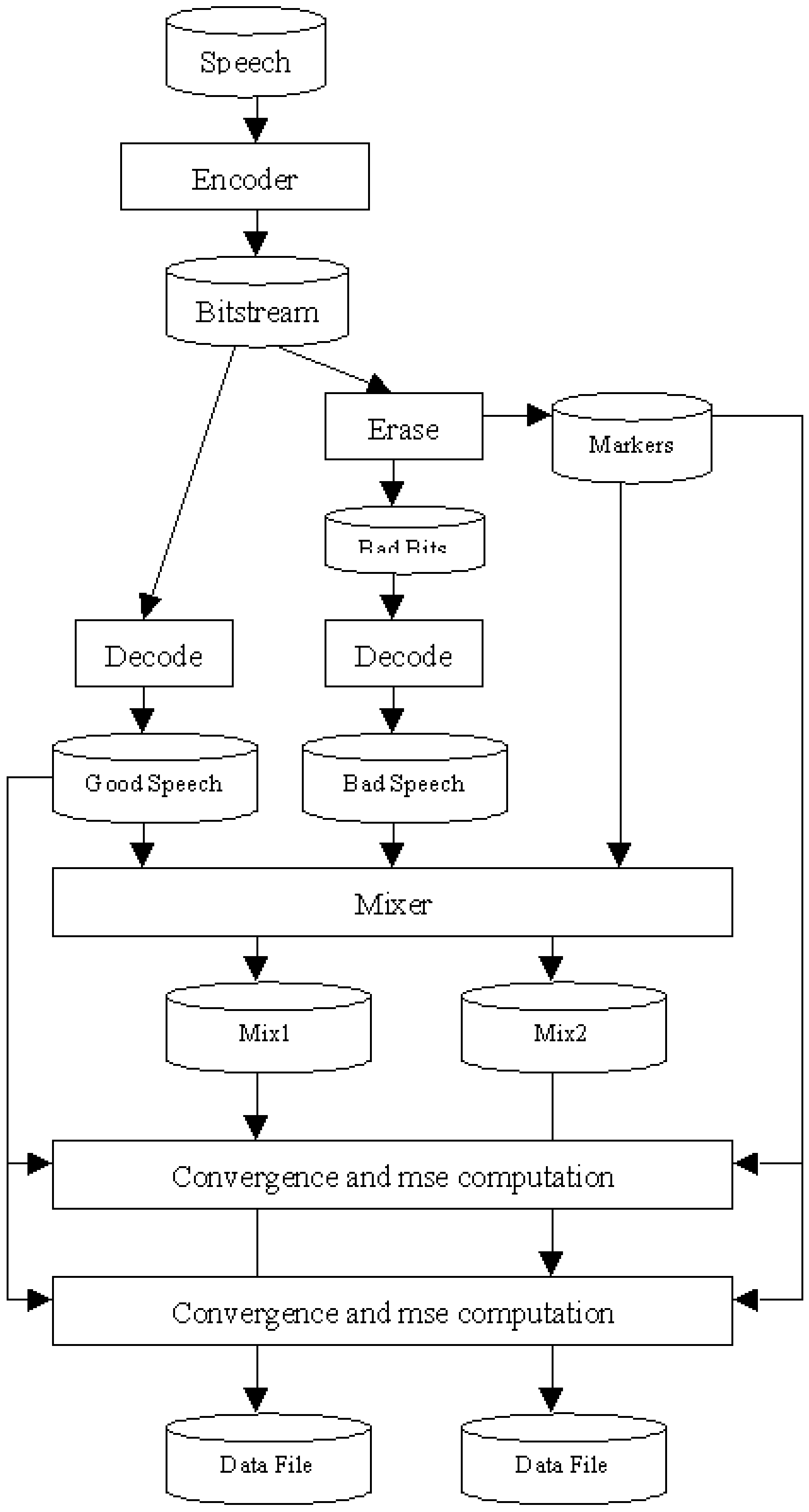

Figure 8: Procedure for comparing speech vs. state restoration

A program called mixer was written. This program accepts two files as input, along with the marker file. It then generates two files as output. Each one has a frame taken from either the first or second input file, depending on the value of the marker for that frame. These two speech files, mix1 and mix2, are then compared against the error free decoded speech, using the mse program as above. The result is the average mse and convergence times for each of the two. The experiment was run using a tcl script called runswitch.tcl. It takes the burstsize as input, and runs the experiment on all of the bitstream files it finds in a directory. Like runtest.tcl, it appends all of the resulting datafiles together, and generates a histogram of the mse's and convergence times from the result.

Not surprisingly, the convergence time for mix1 is zero (since it has its state restored, there is no convergence required), and for mix2, it is the same as given in Table 1 (since restoring the missing speech content has no impact on convergence times).

The average mse's are interesting, however. In the case of mix1, the average mse is measured only over the consecutively lost and concealed frames, and in the case of mix2, over the convergence period. The average MSE and its standard deviation are given in Table 4.

Burst Size	Mix1		Mix2	
	Average MSE	SD of MSE	Average MSE	SD of MSE
1	1.8×10^6	4.5×10^6	1.1×10^6	2.4×10^6
2	2.8×10^6	4.9×10^6	1.6×10^6	2.9×10^6
3	3.0×10^6	4.8×10^6	1.6×10^6	3.1×10^6
4	3.5×10^6	6.3×10^6	1.2×10^6	2.2×10^6
5	3.5×10^6	7.2×10^6	1.0×10^6	1.6×10^6

Table 4: Avg. and S.D of MSE for Mix1 and Mix2

Mix1, therefore, has a larger energy in its error signal, although the duration of the error is much reduced. Mix2 has less energy in its error signal, but the error persists for longer. Which of the two is actually "better" was determined by a subjective evaluation. In the evaluation, each of the 12 subjects listened to a number of pairs of speech signals. Each pair was the same content, but one was the mix1 version (state restoration), and the other was the mix2 version (speech restoration). Thirteen pairs of speech signals were presented; 1 pair (f15.d) contained instances of single frame erasures. The remaining twelve consisted of the speech signals f15.d, f26.d, and m29.d, with 2, 3, 4 and 5 frame erasures each. For each pair, the subject selected which one sounded better, or chose neither if they sounded the same.

The results are depicted in Figure 9. For each speech sample used, the number of subjects who preferred mix1, mix2, or neither is indicated. As the results show, mix1 was preferred more than mix2 for all but two of the twelve speech samples. For the f15.d speech segment (the longest of the three segments), the subjects almost unanimously chose mix1.

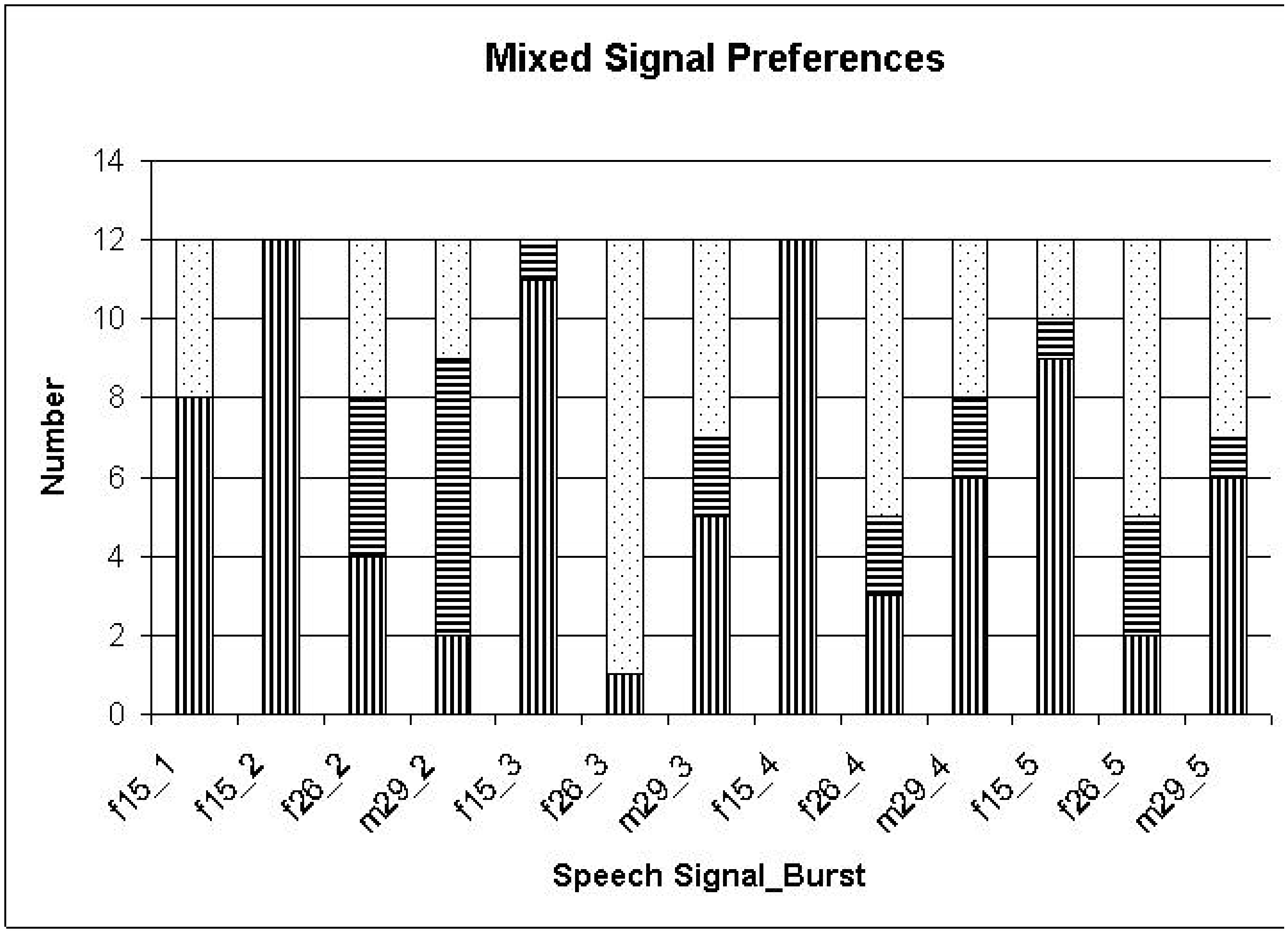

Figure 9: Preferences for Mix1, Mix2, or Neither

You can click here to take this test yourself.

The conclusion would appear to be that restoration of state is more important than just restoring the speech signal for the erased frames.

Summary

In this memorandum, issues of error recovery and resiliency for the G.729 speech coder were explored. After an introduction to the technologies used by G.729, various approaches to error recovery were discussed. The efficacy of many of these solutions depends strongly on the performance of the speech coder in concealing frame erasures. Three particular behaviors were examined. First, the convergence times for resynchronizing encoder and decoder state were measured. The results indicated that these convergence times do not depend strongly on the number of consecutively lost frame erasures. Furthermore, the convergence times are roughly 70 to 100ms in duration, which is shorter than typical network round trip times. This eliminates resynchronization protocols from being useful for G.729. The second behavior examined was the energy in the error signal between the decoded errored bitstream and the decoded unerrored bitstream. The average energy was found to increase with increasing numbers of consecutively erased frames. This increase was sharp when the number of consecutive erasures jumped from one to two, and more gradually increasing from there. Subjective evaluations confirmed this, and demonstrated that the G.729 concealment algorithm works well for single frame erasures, but not more. Finally, the relative importance of state or speech restoration was explored. Subjective evaluations confirmed that it is more important to restore the state of the decoder after a frame erasure than to attempt to restore the speech that was lost during the frame erasures.

Future Work

There are several directions for future work. First, other error criteria need to be explored, falling more in line with established results in the field. Secondly, it is critical to determine which pieces of state are most important to recover after a frame erasure. This will help in designing an actual error recovery mechanism. Thirdly, the subjective tests need to be made more formal, and their scope expanded.

Acknowledgements

The author would like to thank Peter Kroon and Sean Ramprashad for their explanations of coder operation, insight, and G.729 source code. I would also like to thank Henning Schulzrinne for his continued support, along with the members of the High Speed Networks Research Department for their participation in the subjective evaluations.

References

[1] S. Deering, D. Estrin, D. Farinacci, M.H.A. Helmy, V. Jacobson, C. Liu, P. Sharma, D. Thaler, L. Wei, "Protocol Independent Multicast - Sparse Mode: Motivation and Architecture", Internet Draft, Internet Engineering Task Force. Work in Progress.

[2] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real Time Applications", RFC 1889

[3] R. Braden, L. Zhang, S. Berson, "Resource Reservation Protocol (RSVP) -- Version 1 Functional Specification", Internet Draft, Internet Engineering Task Force. Work in Progress.

[4] ITU-T Recommendation G.729 - "Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP)", March 1996

[5] H. Schulzrinne, "Re-Engineering the Telephone Network", IEEE Singapore International Conference on Networks, April 1997

[6] W. Kleijn, K. Paliwal, "Speech Coding and Synthesis", Elsevier Press, Amsterdam, 1995

[7] ITU-T Recommendation G.729 Annex B - "A silence compression scheme for G.729 optimized for terminals conforming to Recomendation V.70", November 1996

[8] J-C. Bolot, "End-to-end packet delay and loss behavior in the Internet", Proc. ACM Sigcomm '93, pp. 289-298, San Fransisco, CA, Sept. 93.

[9] D. Sanghi, A. Arawala, O Gudmundsson, B. Jain, "Experimental Assessment of End-to-End Behavior in the Internet", IEEE Infocom 1993, San Francisco, CA, Mar. 30 - April 1. 1993, pp. 867-874

[10] J-C. Bolot, A. Vega Garcia, "Control mechanisms for packet audio in the Internet", IEEE Infocom '96, San Fransisco, CA, March 1996.

[11] N. Shacham, P. McKenney, "Packet Recovery in High Speed Networks Using Coding and Buffer Management", Proceedings of Infocom 1990, pg. 124-131

[12] V. Hardman, M. Sasse, M. Handley, A. Watson, "Reliable Audio for Use over the Internet", Proceedings of INET '95

[13] J. Rosenberg, "Reliability Enhancements to NeVoT", Columbia University Class Report, December 1996.