A STUDY OF SPEECH/AUDIO CODING ON PACKET SWITCHED NETWORKS

by

Matthew George Podolsky

Memorandum No. UCB/ERL M96/96

19 December 1996

A STUDY OF SPEECH/AUDIO CODING ON PACKET SWITCHED NETWORKS

by

Matthew George Podolsky

Memorandum No. UCB/ERL M96/96

19 December 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Acknowledgments

I would like to acknowledge all of the people who helped make this work possible. I thank my advisor, Martin Vetterli, for his support, advice, and encouragement. I also gratefully acknowledge the support and help Steven McCanne has given me throughout the course of my research.

I would like to thank the following people for their invaluable assistance and discussions, and for making mine an enjoyable working environment: Vivek Goyal, John Haddon, Mike Goodwin, Joe Yeh, Grace Chang, and Francis Ng. I would like to especially thank Mike, Vivek, and Steve for their invaluable feedback and proofreading of this work. I heartily thank Karl Petty, for answering answering numerous scripting language questions, and for giving me the software tools to make full utilization of idle workstations everywhere. I also thank Steve Burgett for his never-ending answers to my never-ending Unix questions.

Lastly, I would like to thank my family for their endless love and support over all of the years.

Contents

List of Figures vi

List of Tables vii

1 Introduction 1

2 A Pyramid Coder For Audio 3
- 2.1 Background and Motivation . . . 5
- 2.2 Coder Structure . . . 8
- 2.3 SNR as a Performance Measure . . . 10
- 2.4 Coder Design . . . 11
  - 2.4.1 Quantizer Choice . . . 12
  - 2.4.2 Filter Choice . . . 15
  - 2.4.3 Transform Coding of the Coarsest Layer . . . 18
  - 2.4.4 Entropy Coding . . . 21
- 2.5 Coder Performance for a Fixed Rate . . . 23
- 2.6 Future Work . . . 26

3 A Network Model For Simulating Audio Over The Internet 28
- 3.1 Source Model . . . 28
- 3.2 Queuing Discipline and Loss Mechanism . . . 31
- 3.3 Network Simulation and Results . . . 32
  - 3.3.1 Description of the Simulation . . . 32
  - 3.3.2 Simulation Results . . . 33
- 3.4 Future Work . . . 36

4 Joint Source/Channel Coding of Audio 38
- 4.1 Pyramid Coder Design and Performance for Lossy Networks . . . 38
  - 4.1.1 Coder Parameters . . . 38
  - 4.1.2 Computing the Distortion . . . 39
  - 4.1.3 Performance Results . . . 40
- 4.2 Redundancy versus Network Congestion . . . 43
- 4.3 Future Work . . . 46

5 Conclusions 48

A Glossary of Abbreviations and Variables 50

Bibliography 53

List of Figures

2.1 Structure of a single-stage pyramid coder. 6
2.2 Structure of a two-stage pyramid encoder. 9
2.3 Structure of a two-stage pyramid decoder. 10
2.4 Effects of quantizer choice on pyramid coder performance. 16
2.5 Effects of filter choice on pyramid coder performance. 17
2.6 Effects of filter length on pyramid coder performance. 18
2.7 Diagram of LPC-based forward and inverse prediction. 20
2.8 Effects of LPC transform coding on pyramid coder performance. 22
2.9 Effects of entropy coding on pyramid coder performance. 23
2.10 Pyramid coder performance under lossless network conditions. 25

3.1 Single source on/off model. 29
3.2 Markov chain for N-source model. 29
3.3 Network model illustrating a priority-drop queue and sources producing three layers of data. 32
3.4 Average percent packet loss rates for two values of G when a priority drop-tail queue is used. 35
3.5 Packet percent loss rates across layers as a function of G. 36

4.1 Pyramid coder performance as a function of α for fixed $\gamma = 0$ (2-layer coding). 45
4.2 Effects of data redundancy as the amount of redundancy used and percentage of total traffic using redundancy varies. 47

List of Tables

3.1 Simulation parameters. 33
3.2 Comparison of theoretical and experimental packet loss rates for a purely random-drop queue. 34

4.1 Performance as a function G for a pyramid coder using automatic-σ scaling uniform quantizers. 41
4.2 Performance as a function of G for a pyramid coder using maximum range uniform quantizers. 43
4.3 Pyramid coding comparison of priority-drop vs. random-drop transmission 44

Chapter 1

Introduction

It has become increasingly popular to use the Internet to transmit audio in real time. Recent years have seen an explosion in the number of tools enabling people to use the Internet for a variety of audio applications, such as person-to-person conversations, voice conferencing, or radio-like music broadcasts. Though the field of audio processing is fairly mature, few of the audio compression algorithms used by Internet audio applications were designed specifically for packet switched networks. The few that were so designed are generally commercial products using private, proprietary encoding schemes often providing questionable quality. Transmission of audio over lossy packet switched networks is the unifying theme of this work.

This report is organized as follows. Chapter 2 covers the design of an audio coder for packet switched networks such as the Internet. It describes the limitations of existing Internet coding schemes and the benefits of a layered coding scheme. Motivations for our choice of a pyramid coding system are given. We also describe our choice of the signal-to-noise ratio as a performance measurement. We study the various individual elements of a pyramid coding structure and discuss various design choices and their performance in a rate-distortion sense. We also examine the performance and behavior of the coder under fixed-rate conditions.

Chapter 3 discusses network issues. A network system model is developed to study prioritized transmission of audio, and we use this model to simulate transmission of layered audio over a priority transmission network under various states of load. Comparisons with non-priority, random-loss methods are drawn to understand the potential benefits of priority transmission.

The larger issue of joint source/channel coding of audio is discussed in Chapter 4. We combine the results of our network simulations with the design of our pyramid coder. Chapter 4 also looks at the use of redundancy to improve the quality of real-time audio data transmission. Finally, in Chapter 5 we draw conclusions and suggest future directions for this work.

A glossary of commonly used abbreviations and variables is given in the appendix.

Chapter 2

A Pyramid Coder For Audio

In recent years, the Internet has become increasingly used to transmit audio in "real-time." While the last two decades have produced a wide variety of audio coders, only some of the more recent coders have been specifically developed for the Internet. Many traditional coders have been adapted and implemented in Internet audio tools. For example, the "visual audio tool" *vat* supports standard coders such as the European cellular phone standard GSM, Linear Predictive Coding (LPC), and Adaptive Differential Pulse Code Modulation (ADPCM) [14]. These coders all use fixed-rate digital speech algorithms that were previously designed without the Internet in mind. Various coders also exist for encoding general audio. The MUSICAM (Masking-pattern Universal Subband Integrated Coding and Multiplexing) audio coder used in the MPEG-I video standard is a high-quality, high-complexity coder that can provide near-CD quality sound [2, 18]. This coder is widely used for real-time audio transmissions in digital satellite systems, which use a transmission medium significantly different than the Internet.

Existing audio coders are typically adapted for Internet transmission by simply packetizing the data produced by encoders. This process is especially simple for speech coders because they usually segment speech into "frames," commonly lasting about 20 ms. Packets consisting of the data of 1 or more frames can be transmitted across the Internet, reassembled by the receiver(s), and decoded into audio. This scenario assumes that network congestion is low enough that few packets are dropped. If this is not the case, and if the coder is not designed with any packet error protection, then the receiver must substitute something for the audio information contained in missing packets. This substitution is commonly silence, synthetic noise, a repetition of the audio from the last received packet, or an

interpolation of known samples (see [10] or [11] for a discussion of receiver-only techniques). Though these latter methods reduce the degradation in audio quality as compared to silence substitution, they are still limited by the lack of any information about the missing packet. Furthermore, the quality of the reconstructed audio quickly decreases as the number of packets consecutively lost increases.

To avoid replacing the audio of a lost packet with a substitute based only on the audio data successfully received (receiver-only techniques), it is necessary to ensure that some information about the audio corresponding to the packet reaches the receiver. One way of achieving this is to send extra information about one frame's audio data together with a nearby frame's data to help the receiver decode a missing packet. Hardman *et al.* [10] used this redundancy-based approach in the Robust-Audio Tool (RAT). Their scheme starts with frames consisting of either 64 Kbps raw data (PCM) or 32 Kbps encoded data (ADPCM). In addition to this fundamental data, each data frame also contains information about a previous frame that is one or more frames behind the current one. This additional information is produced by encoding the previous frame with a low-rate (4.8 Kbps) LPC algorithm. The benefit of this scheme is that if the prior frame's packet is lost but the current frame's packet gets through, the LPC data can be used to reconstruct an approximation to the lost frame's audio. Disadvantages include the single-packet-delay required to recover from the redundant information, and the redundancy's bit-rate overhead, which is 7.5% for PCM- and 15% for ADPCM-based transmission schemes. The process of adding this extra information is a rudimentary form of joint source/channel coding, since the redundant LPC information protecting against channel errors must be coded from the original rather than coded source data.

Rather than add redundant data about a current frame to a future frame, another joint source/channel coding technique splits the data within a frame into two or more layers, and encodes them with different priorities reflecting their relative importance. This priority encoding can come in many forms, for example: adding error-correcting codes to high priority layers; increasing the redundancy of a priority layer by decreasing its compression (for example, if the time samples of one cycle of a sine wave are transmitted instead of only transmitting its frequency and amplitude, individual bit errors among the time samples are less catastrophic to a reconstruction than errors in the frequency or amplitude); or prioritizing the layers and sending them across a network with different qualities of service (QOS). These are only a few examples of types of joint source/channel techniques for layered

data. The first two are better suited for applications where channel errors come in the form of individual bit errors. The latter method is well suited for packet switched networks in which channel errors take the form of packet losses. The Internet is such a network, but unfortunately it currently is not capable of providing different qualities of service.

Even without such a service, layered transmission is still possible. For example, the Priority Encoding Transmission (PET) system uses a multilevel forward-error-correction scheme to provide varying degrees of protection to priority-encoded data [1]. By segmenting the transmission data, as a layered coder automatically does, one can use PET to provide different degrees of packet protection to the different data segments. Another use for a layered encoding over the Internet is the receiver-driven layered multicast (RLM). RLM combines a layered video compression algorithm with a layered transmission scheme to allow receivers of a multicast transmission to individually adapt the local transmission rate according to local network conditions. This scheme could be applied to a layered audio coder to allow receiver-driven adaptation of audio signals.

To utilize schemes such as PET or RLM, an audio coder that produces a layered hierarchy of signals is needed. We consider the application of pyramid coding to audio signals to produce such a layered hierarchy. Section 2.1 introduces and motivates pyramid coding. Section 2.2 explains the structure of the pyramid coder we studied. Our choice of signal-to-noise ratio (SNR) as a performance measurement is discussed in Section 2.3. Section 2.4 describes the design of the individual components of the pyramid coder and compares choices for different components under the framework of rate-distortion curves. Section 2.5 evaluates the coder's performance for a fixed rate as other parameters vary. Finally, future directions to for improving and implementing a pyramid coding scheme are discussed in Section 2.6.

2.1 Background and Motivation

Pyramid coding is a technique that is frequently employed in image coding and computer vision. First introduced by Burt and Adelson [3], pyramid coding consists of an encoder which derives a "coarse" low-resolution version of a signal, predicts the original signal based on that coarse signal, and then takes the difference between the prediction and the original. The coarse approximation and the error residual are the only signals that need to be transmitted: the decoder reconstructs the signal by generating the prediction from

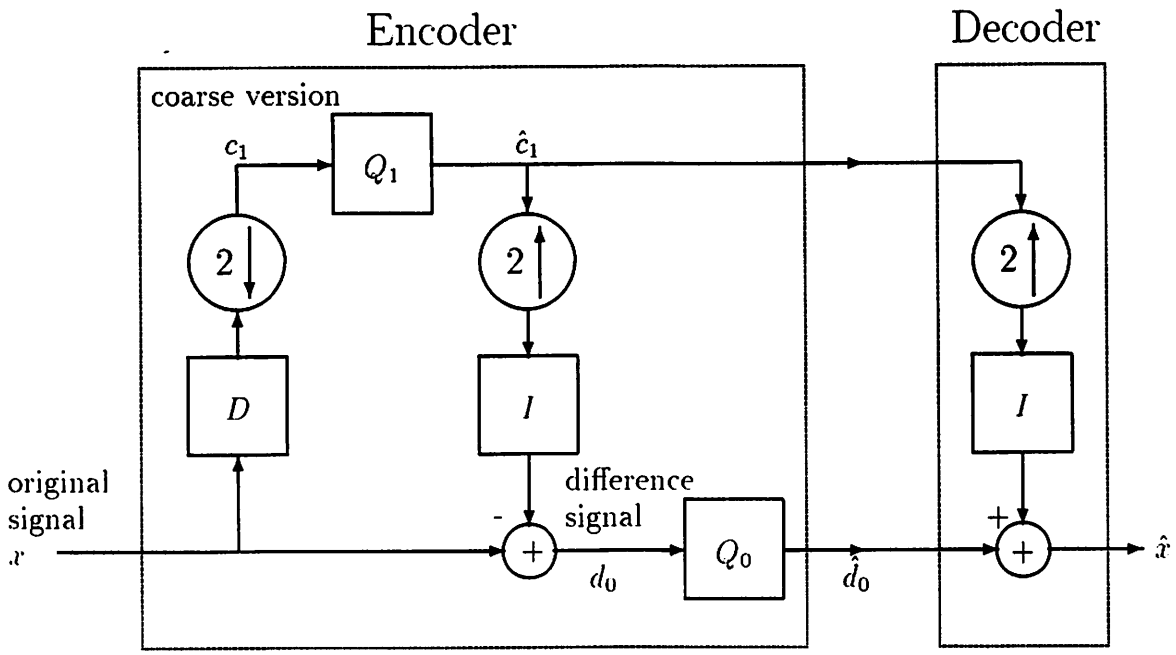

Figure 2.1: Structure of a single-stage pyramid coder. Subscripts refer to layer numbers.

the coarse signal, and adding the residual back to the prediction. Figure 2.1 illustrates a single-stage pyramid coding structure. D and I represent pre-decimation and interpolation operators, respectively. For purposes of notation, the half-rate coarse approximation c_1 is termed the Layer 1 signal, and the full-rate difference signal d_0 is the Layer 0 signal. The structure can be iterated an arbitrary number J times on the coarsest signal to generate a $1/2^J$-rate coarse signal at Layer J, and J difference signals at Layers $J-1$ through 0. The difference signal of Layer i, $i = 0, \ldots, J-1$, is at $1/2^i$ the rate of the input signal x.

A significant advantage of pyramid coding is that there are no constraints on the interpolation and decimation operators. Unlike subband coding, where filters must meet strict orthogonality or bi-orthogonality conditions to achieve perfect reconstruction, operators in pyramid coding can be linear or non-linear filters, median filters, *etc.* In practice they are typically zero-phase FIR anti-aliasing and interpolation filters [25], but as long as the same interpolation operator is used at the encoder and decoder, perfect reconstruction can be achieved in the absence of quantization.

Another advantage of pyramid coding is that it automatically encodes the input in a

hierarchy of layers, assuming that D and I are intelligently chosen. For audio, this means that the predictions based on the coarse signals should sound better than the difference signals. For example, in Figure 2.1, if D and I are lowpass filters, then the prediction based on c_1 will be a lowpass version of the input x, and hence the difference signal d_0 will be a highpass version of x. If x contained more important information in high rather than low frequencies, then d_0 would likely sound better than the prediction. However, if D and I had been chosen to be highpass filters, the prediction signal would have been a better approximation to x than the difference signal, and as such c_1 would have higher priority than d_0, as we desire. For significantly varying signals we could either use adaptive D and I operators to capture the most important information, or use fixed D and I operators and adapt the layers' relative priorities. Assuming an appropriate choice of the D and I operators, for a coder with J iterations the (lowest-rate) coarse signal at Layer J is the highest-priority for reconstructing the original. The second most critical signal is the (second-lowest rate) difference signal at Layer $J-1$, which is added to the coarse signal to produce a new coarse signal at Layer $J-1$. This process repeats until one produces the lowest priority, full rate difference signal at Layer 0. No analysis of the input signal needs to be performed to determine the priorities of the layers—their order always remains the same. Also, the ability to iterate the coder an arbitrary J times means that various operating rates can be achieved by only using the signals of Layers J through i, where $i = (J, J-1, \ldots, 0)$.

Pyramid coders also have the property that if there are no transmission errors, the only distortion results from the quantization of the final difference signal, d_0. In Figure 2.1, if we let x_p be the signal formed by upsampling and interpolating $\hat{c}_1$, then $d_0 = x - x_p$. If we let $\epsilon_d = \hat{d}_0 - d_0$ represent the error due to quantizing d_0, then the reconstructed signal is

$$\hat{x} = x_p + \hat{d}_0 = (x - d_0) + (d_0 + \epsilon_d) = x + \epsilon_d, \tag{2.1}$$

so the only error in the reconstruction is the quantization error ϵ_d. This does not mean that Q_1 can be chosen poorly without penalty. Since Q_1 affects the prediction x_p, a poor choice for Q_1 will increase the power of d_0, making it more difficult to quantize. Furthermore, we are interested in using a priority transmission scheme, so that $\hat{c}_1$ has a high chance of getting through even if $\hat{d}_0$ does not make it: a poor choice of Q_1 means that x_p will serve as a poor reconstruction if $\hat{d}_0$ is lost.

One last advantage of pyramid coding lies in its simplicity. Most of its computational requirements come from the filtering and quantization operations. It can also be recursively implemented to produce a large number of layers. Therefore, a pyramid coding scheme can be easily implemented in real-time with low complexity. The performance benefits of more advanced design elements, such as transform coders or adaptive filters, can be weighed against their added computational complexity.

The primary drawback of pyramid coding is that it results in an oversampled representation of a signal. A one-stage pyramid encoding of an input signal consists of a reduced-rate coarse signal and a full-rate error residual (at the same rate as the original), for a total rate greater than that of the original. The multiplicative increase in rate of this oversampled representation is given below for an L-layer pyramid coder operating on N dimensional signals [25]:

$$s = \sum_{i=0}^{L-1} \left(\frac{1}{2^n}\right)^i < \frac{2^N}{2^N - 1}. \tag{2.2}$$

Though this added overhead quickly decreases as the number of dimensions increases, our interest lies in one-dimensional audio signals, for which the above factor translates to a total rate 50–100% greater than the original. Note that this rate increase refers to the additional rate in a discrete-time, infinite precision pyramid coder. The actual increase in total bit rate can be made less than s through quantization and compression.

2.2 Coder Structure

The structure of a two-stage, three-layer pyramid encoder is shown in Figure 2.2, and the corresponding decoder is shown in Figure 2.3. The encoder transmits the encoded signals s_0, s_1, and s_2. These signals are the outputs of the quantizers; the inverse quantization blocks are shown to emphasize that these signals can be quantizer indices (possibly entropy coded), rather than actual quantization output levels. The encoder structure shown decomposes a signal into a quarter-rate coarse approximation c_2 at Layer 2, and half- and full-rate error residuals, d_1 and d_0, at Layers 1 and 0, respectively. Qualitatively, Layer 2 is the most important to reconstruction, so it is the highest priority signal. The encoder includes both a transform-coding block T which transforms c_2 to y_2, and a trio of quantizers Q_0, Q_1, and Q_2, for the d_0, d_1, and y_2 signals, respectively. The purpose of the transform coder is to make the coarse signal c_2 easier to quantize and compress. Note that the difference signals

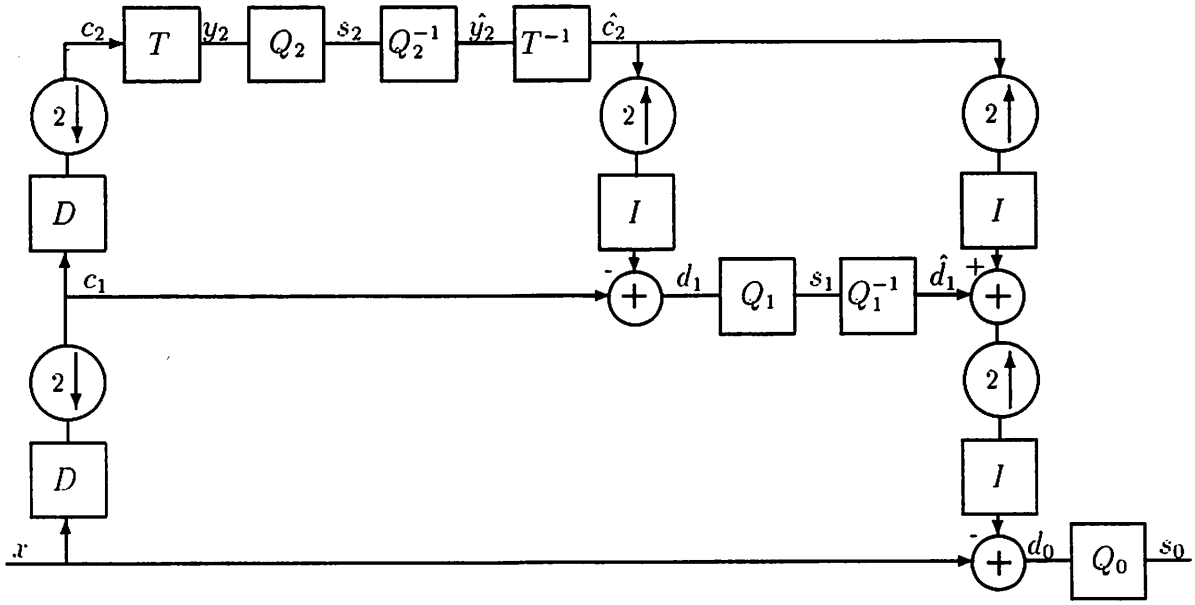

Figure 2.2: Structure of a two-stage pyramid encoder. Subscripts refer to layer numbers. The input signal is x, and transmitted encoded signals are s_i, $i = 0, 1, 2$.

d_1 and d_0 are computed using the quantized signals $\hat{c}_2$ and $\hat{d}_1$, respectively. Because of this, the encoder contains all of the decoding structure, except for the final addition which produces $\hat{x}$. The error residual of Layer L thus accounts not only for the imperfection of the interpolated coarse signal from Layer $L+1$ but also any quantization noise introduced by Q_{L+1}. Assuming that all of the encoded signals s_0, s_1, and s_2 reach the decoder, the only reconstruction error will be due to the quantization of d_0. If no quantization is performed on d_0, perfect reconstruction can still be achieved despite quantization in Layers 1 and 2.

A primary advantage of a pyramid coding scheme is its flexibility. The D, I, Q, and T blocks may all be freely chosen, and they do not have to be the same across every layer. If they are the same across layers, the coder can be implemented recursively and thus easily adapted to produce an arbitrary number of layers. This flexibility, however, hinders attempts to optimize each block, since the underlying signals are jointly dependent. For example, the choices of D, T, Q_2, and I all affect the difference signal d_1, and hence what the optimal Q_1 should be to quantize it. For simplicity, the coder studied was limited to two stages as illustrated in Figures 2.2 and 2.3. Our rationale for this design is discussed

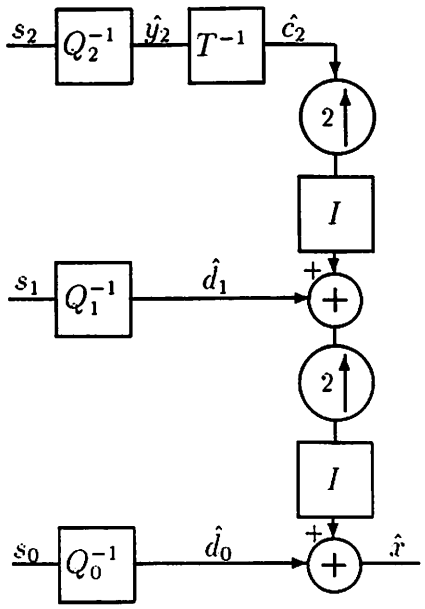

Figure 2.3: Structure of a two-stage pyramid decoder. Subscripts refer to layer numbers. Received signals are s_i, $i = 0, 1, 2$, and the decoded output signal is $\hat{x}$.

in Section 2.4.

2.3 SNR as a Performance Measure

As mentioned in Chapter 1, we chose to use signal-to-noise ratio (SNR) to design our coder and measure its performance:

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum_n x(n)^2}{\sum_n (x(n) - \hat{x}(n))^2} \right). \tag{2.3}$$

This metric is a log ratio of the original signal's power to the error power (mean squared error). The resultant quantity is measured in decibels (dB).

The advantages of using SNR as a performance measure are that it is easy to compute and it provides a quantitative measurement. Consequently, it is well suited when a large number of experimental trials are performed. Unfortunately, it is a poor estimator of subjective quality for a broad range of distortions [24]. For example, substituting white noise instead of silence for the audio of a lost packet will almost certainly increase the error MSE, but the result can sound far better to human listeners [27]. Other objective error measures were considered, including spectral distance, variations of SNR (such as segmental

SNR, granular segmental SNR, and frequency weighted segmental SNR), and the Speech Transmission Index. These were rejected due to their higher complexity [20]. "Segmental SNR" measures the SNR over short segments of long audio signal, and sums all constituent segments:

$$\text{SEG SNR} = \frac{10}{M}\sum_{m=0}^{M-1}\log_{10}\left(\frac{\sum_{n=Nm}^{Nm+N-1}x(n)^2}{\sum_{n=Nm}^{Nm+N-1}(x(n)-\hat{x}(n))^2}\right). \tag{2.4}$$

N is the segment length and M is the number of segments in the signal. It is not inherently much more complex than the classical SNR, but it suffers from the problem that if there are intervals of silence where the signal is of very low level, even a small amount of noise will produce a large negative SNR in that interval [20]. To effectively use segmental SNR one needs to identify silent segments and exclude them from the measurement, which leads to a more complex calculation.

Subjective measures involving human listening, such as mean opinion score (MOS) tests, provide the ultimately accurate measure of audio quality but are time consuming, difficult to arrange, and not practical when a large number of experiments are performed.

For our experiments we resorted to classical SNR. Though a suboptimal performance measure, it is cheap and easy to compute. Using SNR allowed us to vary a large number of parameters in a reasonable amount of time, and thus explore a large number of design issues and network interactions. In future work we hope to apply more significant performance measures, such as those mentioned above, to our experimental design.

2.4 Coder Design

This section discusses the experiments and choices made in the design of the coder in Figures 2.2 and 2.3. Rate-distortion (R-D) curves are used to make comparisons between different design choices. These curves were generated by picking a set of coder parameters and varying the number of bits allocated to each quantizer (Q_0, Q_1, and Q_2) between 0 and 8 to vary the total rate. This led to a total of $9^3 = 729$ data points per design. All of these data points were used to create an initial scatter plot of MSE versus bit-rate. The data points falling on the convex hull of this plot were then transformed into SNR values and connected to form an R-D curve. Since the performance measure is on a logarithmic (dB) scale, the R-D curves are not necessarily concave, but they would be on a linear curve. Though a specific point along the R-D curve might not be achievable, the curves give an

idea of the overall performance of the coder. Due to the different sample rates of the levels, the overall number of bits b spent per input samples is

$$b = \sum_{i=0}^{2} \frac{b_i}{2^i} = b_0 + b_1/2 + b_2/4, \tag{2.5}$$

where b_i is the number of bits used quantize Layer i. A choice of $b_i = 0$ corresponds to not coding Layer i at all. We considered only the coder's performance on speech signals. Design issues that would be different for general audio are noted within each section below. The data used to produce the plots was 22.000 16-bit-linear samples of a speech signal sampled at 8 Kilo-samples/second (Ksps), so that bit allocations using b bits per input sample correspond to a total rate of $8000b$ bits/second (bps). This relatively small data set was chosen to capture the relative behavior of different design choices, not to produce absolute performance numbers. A broader and larger set of data is used to produce more accurate SNR numbers in Section 2.5. Because the input data is 16-bit, for $b < 16$ we always expect some distortion in the output due to quantization.

2.4.1 Quantizer Choice

The Problems Posed By Pyramid Coding

The choice of quantizers and the number of bits to allocate to them is perhaps the trickiest part of pyramid coding. Unlike other schemes, the quantizers used in pyramid coding cannot be designed independently to achieve minimum distortion. Consequently, quantizer design methods such as the Lloyd-Max algorithm [8] cannot be easily employed. The choice of a quantizer at layer L will affect the prediction $\hat{x}_{p_{L-1}}$, and thus in turn the difference signal $x_{d_{L-1}}$, which is then quantized by Q_{L-1}. This process repeats, and so the quantizer at layer L affects the signals input to the quantizers at all the layers below L, namely 0 through $L-1$. A dependent quantization scheme (such as general vector quantization (VQ) of all of the signals) might produce the minimum distortion, but would be undesirable because it eliminates the layered structure inherent in pyramid coding. Hence, if we want a pyramid scheme that produces separable layers, we are limited to using individual scalar quantizers for each layer. Note that these quantizers can still be dependent upon each other, as in hierarchical VQ, as long as they produce separate symbols for each layer.

Even with the restriction of scalar quantizers, the search for optimal quantizers still grows exponentially as the number of layers increases. If we have K different quantizers (for example, uniform quantizers using 1 through K bits) at each layer, and we have L layers, in the worst case we will need to search through K^L combinations of L-tuples of quantizers. Even if this total K^L is not so large as to exclude an exhaustive search, such a search is still hampered by the need to include the effects of the pyramid coder in the search process. For example, given the coder of Figure 2.2, it might be desired to find the optimal quantizers of arbitrary bit allocations (for example, between 1 and 8 bits) for each layer. There are 3 layers, and hence 512 possibilities to explore. If we were to use the Lloyd-Max algorithm, we would have to go through the following process:

1. Using a set of training data for x, create training data for c_1 and for y_2 using the relevant part of the encoding structure of Figure 2.2 . Using the y_2 training data, apply the Lloyd-Max algorithm to design 8 quantizers Q_{2_i}, which correspond to allocating $i = (1, \ldots, 8)$ bits to Q_2.

2. Quantize the y_2 training data using each of the 8 Q_{2_i} to produce 8 training signals $\hat{y}_{2_i}$, $i = (1, \ldots, 8)$. Upsample and interpolate each of these signals and take their difference with the training data c_1 to produce 8 training signals d_{1_i} for the Layer 1 difference signal.

3. Using Lloyd-Max, for each training vector d_{1_i}, $i = (1, \ldots, 8)$, design 8 quantizers $Q_{1_{i,j}}$ using j bits, $j = (1, \ldots, 8)$. We now have a set of 64 quantizers for d_1.

4. Quantize each of the d_{1_i} training signals with each of the $Q_{1_{i,j}}$ quantizers to produce 64 training signals $\hat{d}_{1_{i,j}}$, $i, j = (1, \ldots, 8)$. Upsample and interpolate each of these and take their difference with the training data for x to obtain 64 training signals $d_{0_{i,j}}$ for d_0.

5. Using Lloyd-Max, for each training vector $d_{0_{i,j}}$, $i, j = (1, \ldots, 8)$, design 8 quantizers $Q_{1_{i,j,k}}$ using k bits, $k = (1, \ldots, 8)$. We thus produce a total of 512 quantizers $Q_{0_{i,j,k}}$ for d_0, $i, j, k = (1, \ldots, 8)$.

We are left with 8 optimal quantizers for y_2, 64 for d_1, and 512 for d_0, so that for a choice of (i, j, k) bits to quantize Layers (2, 1, 0), respectively, we would use $(Q_{2_i}, Q_{1_{i,j}}, Q_{2_{i,j,k}})$ to optimally quantize the pyramid coding signals. We therefore conclude that designing optimal

quantizers for pyramid coding is a difficult and tedious process, even when other parameters of the coder are fixed. As the rest of the coder's design parameters (*e.g.* interpolation filters, transform coders) are varied, optimality would require the quantizer design procedure above to be repeated for each variation of a parameter. Due to practical limitations, we restricted our design search to suboptimal quantizers whose behavior could be easily defined for any given allocations of bits. This decision led to the use of uniform quantizers for our study. We also restricted ourselves to using the same quantizer design method across all three layers (*i.e.* we did not try combining one type of quantizer for one layer with a different type for a another layer). A discussion of more efficient ways to select dependent quantizers in a pyramid coding setting can be found in [22].

Uniform Quantizers Studied

We studied three different types of symmetric uniform midrise quantizers, each employing different methods of determining the maximum range of input values that can be quantized without overload. The overload region determines the range of values that fall further than $Q_{\text{step}}/2$ from an output level, where Q_{step} is the quantization step size. Following is a description of the three types of quantizers studied, and the abbreviations used to refer to them in figures:

- Maximum scaling quantizers. For each layer, the quantization range is chosen so that no input level overloads the quantizer. Abbreviated "Max".
- 4-σ quantizers. The overload regions are determined according to the common "4-σ" rule, which states that the quantization range should be chosen so that input levels in the range $[-4\sigma, 4\sigma]$ do not overload the quantizer, where σ refers to the input variance. Abbreviated "4-s".
- Automatic-σ scaling quantizers. These quantizers use a simple heuristic to determine the overload region according to not only σ but also b_i, the number of quantization bits used for Layer i. The quantization range is chosen so that input levels in the range $[-\sigma b_i/2, \sigma b_i/2]$ do not overload the quantizer. Abbreviated "A-s".

The automatic-σ quantizer overload region rule was chosen after studying data showing optimal choice of overload regions versus quantization levels for Laplacian, Gamma, and Gaussian densities. Note that the given formula is equivalent to the 4σ rule when $b_i = 8$.

Figure 2.4 compares the R-D curves obtained when using each of the above quantizers in conjunction with length-19 windowed sinc filters for D and I, in the absence of transform coding. The figure shows that for bit rates below 70 Kbps, the automatic-σ scaling quantizers perform much better than the the other two quantizers, which have fixed overload regions independent of the number of bits. It is also notable that the SNRs of these fixed overload quantizers go negative at low bit-rates, while the automatic-σ quantizer's SNR does not. Recall that all three designs are symmetric *midrise* quantizers, so zero is not an output value. The fixed overload region quantizers produce negative SNRs because as they use fewer quantization bits, their quantization step size increases faster than the automatic-σ's does, and so their quantizer output levels closest to zero may still be of greater magnitude than a majority of input signal values, which lie closer to zero. This coarse quantization can result in a large increase in the energy of the quantized signal, eventually leading to a reconstruction $\hat{x}$ with so much more power than x that the error power is greater than the input power. Hence, a negative SNR results. The automatic-σ quantizers avoid this by sacrificing a larger maximum possible quantization error for a smaller quantization step size, which can more accurately quantize values close to the origin. Note that at higher rates the two fixed-overload quantizers perform better, because their quantization step size decreases with larger bit allocations and they can better quantize low-level signals, so that they can also take advantage of their wider quantization range.

2.4.2 Filter Choice

We performed a brief exploration of different half-band lowpass filters to use for D and I. To start, we decided to restrict ourselves to using the same filter for D and I, and also to use the same filter across layers. This choice makes sense for a high-rate speech signal for which c_2 and c_1 will consist of the lower quarter and half of the speech spectrum, respectively. For our test data, which is 4 KHz speech sampled at 8 Ksps, a reconstruction based solely on c_2 only contains the information in the 0–1 KHz band. This is generally inadequate for intelligibility, but if it is the case that Layer 1 packets are rarely dropped (so that the reconstructed signal is usually calculated from at least Layers 1 and 2), then a brief period of reconstruction based on only c_2 may not be very disturbing to a listener. This is akin to the previously mentioned receiver-only reconstruction techniques, except that the prediction based on c_2 is the information inserted between losses of d_0 and d_1, and c_2 is

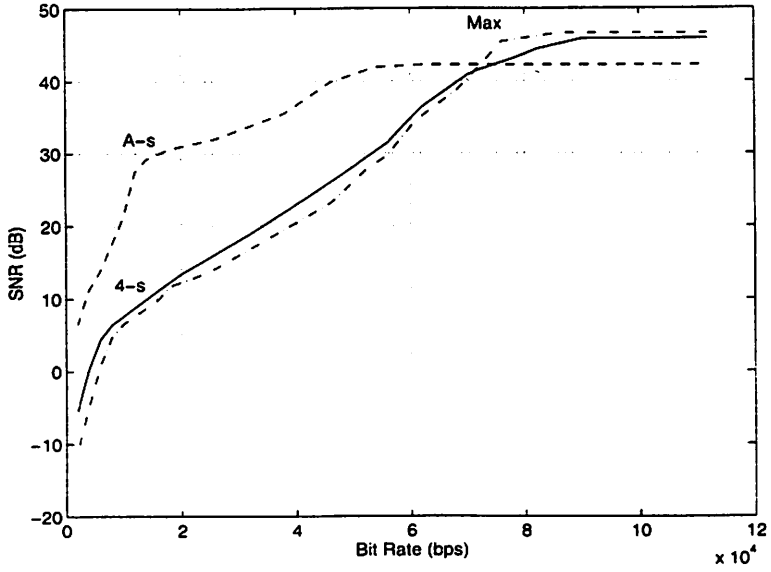

Figure 2.4: Effects of quantizer choice on pyramid coder performance. Labels refer to choice of overload regions for uniform quantizers.

successfully transmitted from the source to the receiver. Unlike receiver-only techniques, however, there is not a complete loss of all data corresponding to a frame of audio because of the priority encoding of the data.

If we were to encode general audio, the most significant information is not necessarily limited to the lower half of the bands. For a general audio encoding it may be better to take the middle of the input spectrum, or to adapt the filters according to the information content in the signal. For example, we could use a 32-band analysis as in MUSICAM to determine what bands contain the most critical information. We could then determine the 16 contiguous bands (taking up half of the input spectrum) containing the most information, and use a cosine-modulated filter [25] for D_0 (the decimation filter between Layers 0 and 1) to extract this portion of the spectrum and form c_1. We could then apply the same technique to determine D_1, using a cosine-modulated filter bank that filters the 8 contiguous bands (of the original 16 band subset) containing the most information.

The filters we studied were all odd-length, zero-phase, symmetric FIR filters. Because the filters have no phase shift, the prediction signals are aligned with the original signals to minimize the difference signals. The filters were also scaled to unit norm to take into account the reduction in power due to downsampling. We studied three types of lowpass

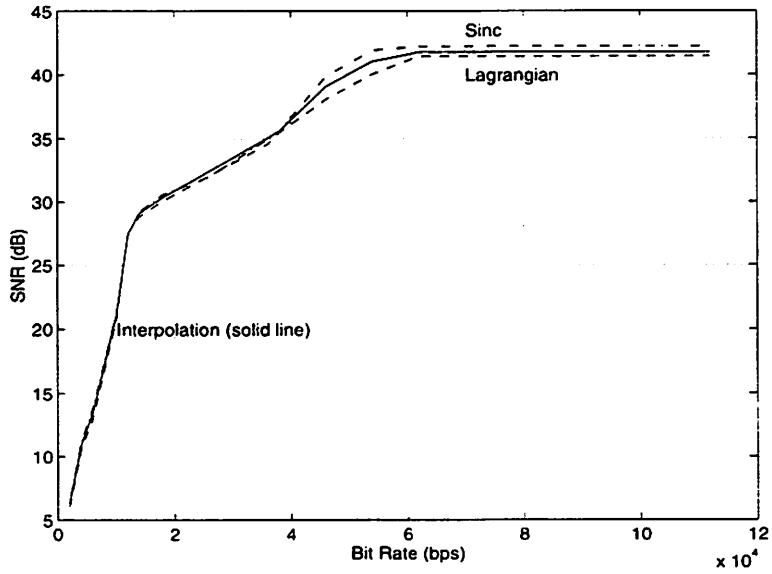

Figure 2.5: Effects of filter choice on pyramid coder performance. Performance of length-19 windowed sinc, Lagrangian, and bandlimited interpolation filters is shown.

filters:

- Half-band sinc filters windowed by a Hanning window.

- Lagrangian polynomial interpolation filters. These filters perform Nth order polynomial interpolation on sequences that alternate zeros with samples values (as obtained by upsampling by 2). For zero-phase, N must be odd: in this case the filter length is $2(N+1)-1$.

- Bandlimited interpolation filters designed using the `intfilt` command in MATLAB.

The filters were all tested using automatic-σ scaling uniform quantizers and without using any transform coding of c_2. Figure 2.5 compares the performance of the above filters when they are all length-19. It can be seen that the sinc filter performs slightly better than the bandlimited interpolation filter, which in turns has slightly higher performance than the Lagrangian polynomial interpolation filter. The relative performance rankings were consistent across filter lengths. The similarity among the R-D curves is not surprising, given that all the filters are variations of a halfband lowpass filter.

We also studied the effect of filter length on performance. Figure 2.6 shows the relative performance of length-7, 11, and 19 Lagrange filters. Performance increases with filter

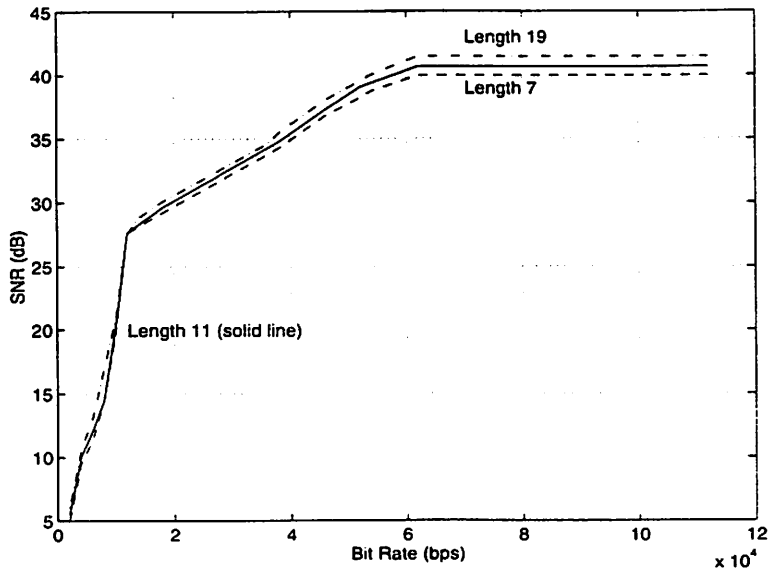

Figure 2.6: Effects of filter length choice on pyramid coder performance. Performance of length-7, 11, and 19 Lagrangian interpolation filters is shown.

length, as expected. The performance gained by increasing the length from 7 to 11 is approximately the same as that gained between going from length-11 and length-19. Varying the length of the two other types of filters produced similar results.

2.4.3 Transform Coding of the Coarsest Layer

We explored transform coding the coarse signal c_2 with the goal of making it better suited to quantization. While transform coding could be performed on any of the difference signals as well, we concentrated on c_2 because it has the most direct relationship to the original signal x. Unlike d_0 or d_1, which are dependent on quantized, up-sampled and filtered signals from higher layers, c_2 is formed by twice filtering and downsampling x. The relationship between c_2 and x can be expressed in terms of their Fourier transforms:

$$C_2\left(e^{j\omega}\right) = \frac{1}{4}\left[D\left(e^{j\frac{\omega}{2}}\right) + D\left(e^{j(\frac{\omega}{2}-\pi)}\right)\right]\left[D\left(e^{j\frac{\omega}{4}}\right)X\left(e^{j\frac{\omega}{4}}\right) + D\left(e^{j(\frac{\omega}{4}-\pi)}\right)X\left(e^{j(\frac{\omega}{4}-\pi)}\right)\right]. \tag{2.6}$$

This relationship is fairly simple: it does not involve any nonlinear effects of quantization. For instance, if D is a half-band lowpass filter, $C_2(e^{j\omega})$ is simply the lower quarter frequency band of $X(e^{j\omega})$. Information about the input x can be used to help determine an appropriate transform. For example, if the input is an 8 KHz speech signal sampled at 16 Ksps, c_2 will

consist of the input's lower 0 to 2 KHz band (and its reflection from 2 to 4 KHz). Since there is a considerable amount of speech information in this band, it is a reasonable assumption that transform methods that work well on wideband speech signals would work well on this filtered signal. For example, linear predictive coding is used in many speech coders to compress speech. The 0 to 2 KHz band of speech still contains a high degree of correlation between samples, so LPC can be applied to try to remove the correlation and reduce the signal's variance. On the other hand, if x is 22-KHz bandlimited music sampled at 44 Ksps, c_2 will contain the 0 to 5.5 KHz band. The type of music x represents will dictate whether most, some, or little of the audio information appears within this band. For this case a subband analysis which splits the 5.5 KHz signal into smaller frequency bands would be a more appropriate transformation than LPC. Compression could be achieved by determining the critical bands containing the highest amounts of acoustic information, similar to the methods used in MUSICAM, and spending a majority of the bit budget on quantizing the high-information bands.

To date, we have studied only linear predictive coding as a transform technique. It is discussed in the following section.

Linear Predictive Coding

Linear predictive coding is a transform-coding technique employed in many standard speech coding algorithms, such as GSM [6] and LPC-10 [23]. A p^{th}-order LPC analysis predicts the current sample of a signal x based on p previous ones. This involves designing a p-tap filter $P(z) = \sum_{k=1}^{p} a_k z^{-k}$ such that the prediction error $E(z) = X(z)[1 - P(z)]$ is minimized with respect to some error measurement, usually MSE. Figure 2.7 shows how the prediction error signal can be used to recreate the input signal by inverting the analysis filter $A(z) = [1 - P(z)]$. In the diagram, the coefficients p_k of $P(z)$ have already been determined via an LPC analysis technique such as the Levinson or Durbin algorithm [4]. Such a system can be used to whiten the error signal, $\epsilon(n)$, so that $\epsilon(n)$ has lower power than $x(n)$ and is thus better suited for quantization. If there is no quantization of $\epsilon(n)$, perfect reconstruction is achieved.

Most speech coders use LPC for more than simply whitening a speech signal. Instead, they rely on the common speech production model [21] which represents the composite

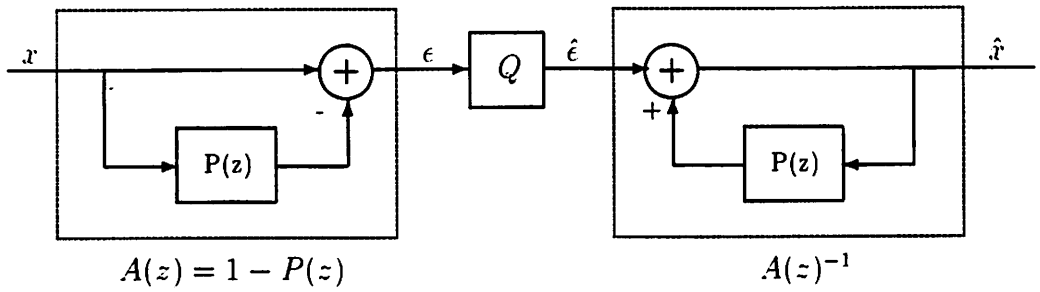

Figure 2.7: The coefficients of $P(z)$ are computed using LPC analysis of the input data.

effects of radiation, vocal tract, and glottal excitation by a time varying filter of the form

$$H(z) = \frac{K}{1 - \sum_{k=1}^{p} a_k z^{-k}}. \tag{2.7}$$

where K is a scalar gain parameter. $H(z)$ can be recognized as having the same form as the inverse prediction filter $1/A(z)$. Speech coders using LPC analysis estimate $A(z)$ so that they can model the speech production system $K/A(z)$. Instead of using LPC to determine a prediction error signal, they attempt to determine the input excitation signal to $H(z)$ that produced the speech signal x. This corresponds to analyzing x to determine K and whether x corresponds to a voiced or unvoiced sound. For unvoiced sounds, white noise is used to excite $1/A(z)$, and for voiced sounds the excitation source is a series of impulses whose spacing corresponds to the estimated pitch of x.

One problem with using this type of LPC analysis in our pyramid coder is that it is a model-based approach that does not attempt to minimize the SNR of the error signal. SNR measures are appropriate only for coding systems that reproduce an estimate of the input signal such that the original and estimate signals can be time aligned and the noise can be accurately calculated [20]. Model-based LPC techniques generally fail to align their estimate with the original input. Furthermore, most existing LPC algorithms have been designed for a single input sampling rate: parameters such as the filter order p may not be appropriate or optimal for other rates. Since our pyramid coding scheme is applicable to any rate input signal, we limited ourselves to exploring the benefits of the system shown in Figure 2.7.

To incorporate this LPC-based whitening system into our coder, we split c_2 up into blocks of length L and computed the p predictor coefficients of $P(z)$ for each block. The LPC analysis to determine the coefficients was implemented using the autocorrelation method

[21]. For this framework, $A(z)$ acts as the time-varying transform block T shown in Figure 2.2, and $A(z)^{-1}$ is the time-varying inverse transform T^{-1}. In our experiments we varied the LPC order p between 3, 5, and 10, and the block length L between 50, 100, and 200. Because c_2 is at one fourth the rate of the input signal rate, these block lengths correspond to 25, 50, and 100 ms length segments of the original signal x, respectively. Using automatic-σ quantizers and length-19 windowed sinc filters, our overall conclusion is that using LPC on c_2 provided negligible performance gain. Performance improvements would require more advanced quantizer design, and/or implementing an adaptive model-based approach which could handle different sample rates.

Among the parameter variations we tried, we found that a block length of 200 samples led to any improvement in SNR, and that this improvement came only at rates above 60 Kbps. For a fixed frame length of 50 or 100, the choice of filter order made no difference. For a frame length of 200, a filter order of 3 provided the best performance gain, followed by 10 and then 5. The LPC analysis process involves a matrix inversion to obtain the predictor coefficients. Our results seem to indicate that frame lengths of 100 samples or less do not provide enough data for the LPC analysis to accurately compute the global minimum-error filter. Similarly, higher order filters 5 and 10 likely result in more poorly conditioned coefficient analysis conditions than the smaller order of 3. Figure 2.8 compares the best combination of values, $p = 3$ and $L = 200$, to doing no LPC coding at all. Note that in computing the rate for the LPC curve we did not take into account the added rate of transmitting the filter coefficients every frame. However, given the small number of coefficients, long frame length, and extensive existing research on efficient quantization of LPC parameters (see [26] or [17] for examples), we felt any additional rate overhead was negligible.

2.4.4 Entropy Coding

We explored the effect of adding entropy coding to the quantizers by calculating the zero-order entropy of the quantized signals, and then substituting this entropy for b_i in Equation 2.5 to recalculate the rate of a given bit allocation. The zero-order entropy gives an estimate of the information content of a signal when the exact distribution of the signal

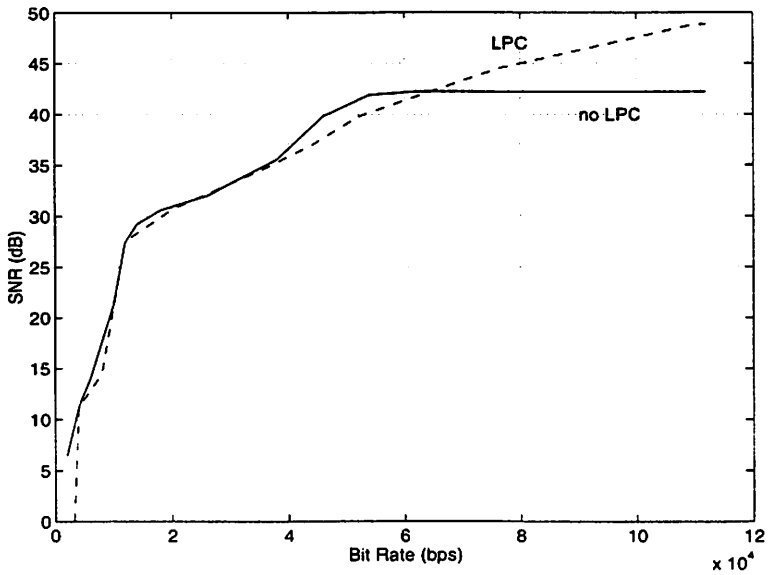

Figure 2.8: Effects of LPC transform coding on pyramid coder performance. Automatic σ-scaling uniform quantizers and a windowed length-19 sinc filter were used to generate the data.

is not known. The zero-order entropy of the output of a B-bit quantizer is formulated as

$$H_0 = -\sum_{i=0}^{2^B-1} \frac{n(i)}{N} \log_2 \left(\frac{n(i)}{N} \right). \tag{2.8}$$

where N is the length of the signal and $n(i)$ is the number of times quantizer Level i is output. The zero-order entropy can be used to give an idea of what the average bit rate needed to encode the quantized signals would be if arithmetic coding on the quantized data was performed.

Figure 2.9 shows the difference between plotting R-D curves using zero-order entropy bit-rate versus raw bit-rate when automatic-σ uniform quantizers are used. Because entropy coding is a lossless process (it does not affect the quantized signals, merely how they are represented), it does not affect the performance for a given (b_0, b_1, b_2) quantizer allocation. The average bit rate needed to transmit the coded signals does change with entropy coding, so that the effect of entropy coding is to compress the rate axis of the R-D curves. Although an entropy code cannot be designed to exactly achieve the zero-order entropy, the figures indicate the potential for significant gain from entropy coding a signal. Other types of filters showed similarly large rate reductions when zero-order entropy rates were calculated.

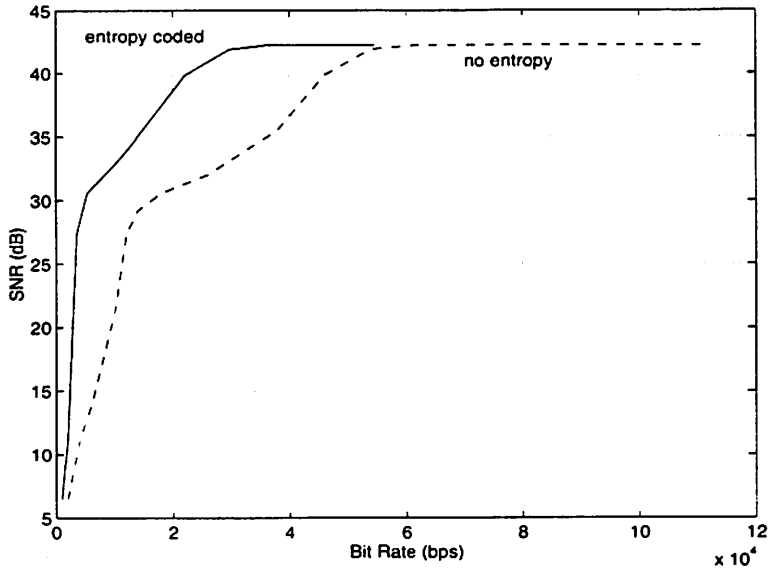

Figure 2.9: Effects of entropy coding on pyramid coder performance. Automatic σ-scaling uniform quantizers and a windowed length-19 sinc filter were used to generate the data.

2.5 Coder Performance for a Fixed Rate

In addition to characterizing our coder's performance across different rates, we also investigated its performance for a fixed rate budget. In other words, if we fix the total number of bits b that we allocate to each input sample, we want to know how the coder's performance changes as we vary the fractions of b allocated to the different layers. For a fixed value of b, the R-D curves give an upper bound on the best performance possible: varying the allocation of bits across the three layers amounts to moving vertically among different operating points at or below the R-D curve. We denote the fraction of b bits allocated to Layers 0, 1 and 2 as α, β and γ, respectively. For a fixed choice of b we have the following constraints on α, β, and γ:

$$\alpha + \beta + \gamma = 1 \tag{2.9}$$

$$(\alpha, \beta, \gamma) \geq 0 \tag{2.10}$$

$$(\alpha b, 2\beta b, 4\gamma b) \in \mathcal{Z}. \tag{2.11}$$

The first constraint expresses that we will spend a total of exactly b bits per input sample on the three quantizers. A result of this is that there are only two free variables: if α and β are given, then γ is completely determined by $\gamma = 1 - (\alpha + \beta)$. The second constraint simply

states that we cannot allocate a negative number of bits to any layer. The third constraint says that the number of bits used by each quantizer must integer valued. If we again let b_i be the number of bits used to quantize Layer i, we have $(b_0, b_1, b_2) = (\alpha b, 2\beta b, 4\gamma b)$. Using these constraints, we see that for a fixed b the possible values for b_i are

$$b_0 = \alpha b \in (0, 1, \ldots, b) \tag{2.12}$$

$$b_1 = \beta b \in (0, 1, \ldots, 2b) \tag{2.13}$$

$$b_2 = \gamma b \in (0, 2, 4, \ldots, 4b). \tag{2.14}$$

The reason b_2 is a multiple of 2 bits is that in terms of total bit rate, spending 1 bit on Layer 2 is equivalent to spending 0.5 bits on Layer 1 or 0.25 bits on Layer 0. Hence there is no way to allocate an integer b bits among the layers if b_2 is odd.[1] The highest layer of a pyramid coding structure is always subject to this constraint. If we had a fourth layer then b_2 could vary in increments of 1 bit, but b_3 would be restricted to even bit values.

Figure 2.10 is a three-dimensional representation of the coder performance as a function of α and β, for b fixed at 8-bits per input sample. 16-bit input is again used, so since $b = 8$ there is no error-free coding for any (b_0, b_1, b_2) allocation. Since γ is completely determined for fixed α, β, and b, we do not need a fourth dimension to view its effect on the results. Different values of γ correspond to different lines of slope -1 in the α-β plane. Using the constraint of Equation 2.9, we can see that $\gamma = 0$ corresponds to the line $\alpha + \beta = 1$ bisecting the portion of the α-β plane shown in Figure 2.10, and corresponds to maximum freedom for choice of α and β. As we increase γ, we decrease the number of possible choices for α and β, and move parallel to the $\alpha + \beta = 1$ line in the direction toward the origin. We stop at the origin, where $\gamma = 1$. The half of the α-β plane on the other side of the $\alpha + \beta = 1$ line (away from the origin) violates the constraint of Equation 2.9 and is not a valid operating range.

We can ascertain certain coder operating characteristics by examining Figure 2.10. First, if we fix β or γ, as we increase α the performance generally improves. This is not surprising, since we saw from Equation 2.1 that the reconstruction error (when no information is lost) is simply the error in quantizing d_0. Of course the choice of Q_1 and Q_2 can effect how well d_0 can be quantized, but clearly by adding bits to b_0 we generally increase the SNR. We can also see that there is a tremendous performance increase when moving

[1] Note that entropy coding could result in fractional (non-integer) rates for the b_i.

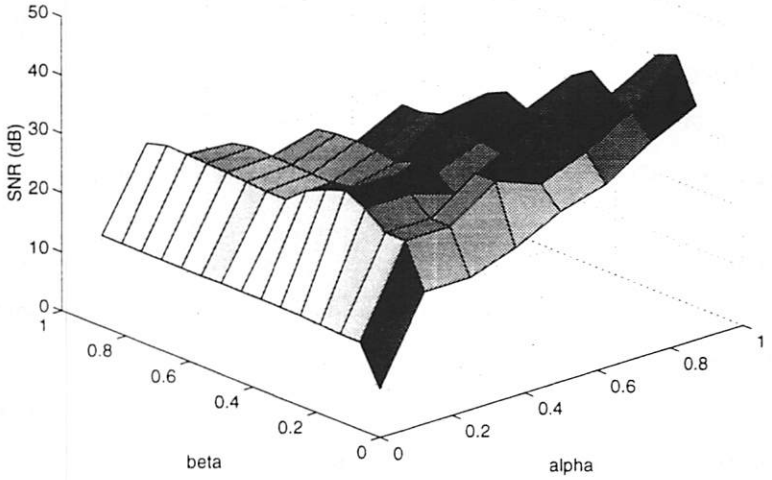

Figure 2.10: Pyramid coder performance under lossless network conditions. Data calculated for $b = 8$ bits/samples, length-19 windowed sinc filters, automatic σ-scaling uniform quantizers, and no transform coding.

from the line $\alpha = 0$ to $\alpha = 0.125$. This corresponds to the gain of using 1-bit to code Layer 0 ($\alpha = 0.125$) over not coding it at all ($\alpha = 0$). The performance improves as α increases, up to a point. This point stops short of $\alpha = 1$, which corresponds to direct quantization of the input x without any pyramid coding producing higher levels. This is indicative of the reduction in variance due to producing a prediction for x and quantizing a difference signal d_0, versus direct quantization of the input x. In fact, the peak of Figure 2.10 occurs at $(\alpha, \beta, \gamma) = (0.875, 0.0625, 0.0625)$, or $(b_0, b_1, b_2) = (7, 1, 2)$, not at $(\alpha, \beta, \gamma) = (1, 0, 0)$, or direct 8-bit quantization of x. This indicates that there is also a gain to be had by coding Layer 2, and thus quantizing Layer 1 as a difference signal. We also see a performance gain in moving from the line $\beta = 0$ to $\beta = 0.0625$, indicating that we do better by quantizing d_1 and producing an intermediate coarse signal $\hat{c}_1$ than we do by basing the full-rate prediction signal for x on the quarter-rate coarse $\hat{c}_0$.

What Figure 2.10 cannot show is how the performance is affected if there are losses in transmission. Since we plan to use a priority-transmission scheme, we can not see the gain of allocating more bits to the higher priority Layers 1 and 2. Figure 2.10 only gives the optimal allocation when no packets are lost. As the network becomes congested and packets are dropped, we expect to allocate more bits to Layers 1 and 2 (since they have a

higher probability of being received), under the constraint that we do not allocate so many bits that Layers 1 or 2 experience frequent losses. We develop a network simulation in Chapter 3 so that we can study this very problem, and we apply the simulation's results to our coder design in Chapter 4.

2.6 Future Work

There is much that can be done to improve the pyramid coder's performance. We believe the biggest improvement in coder performance will come by using better quantizers. The next logical step would be to try to design Lloyd-Max quantizers, or closer approximations to them than the various uniform quantizers we used. We could do this via an exhaustive search, or by applying some of the techniques explained in [22].

A better distortion measurement should also produce a real-world gain in performance. One place to start would be implementing segmental SNR with a threshold to ignore silent regions. This should not only provide a more accurate measure of performance, but should also better take into account the local distortion resulting when a packet is lost. And before a pyramid coder is implemented in a software application, subjective listening tests should also be conducted.

To implement the coder for general audio or music, we should repeat the testing framework of Section 2.2 on these kinds of signals. In particular, this would likely result in designing different filters, possibly time-varying, as described in Section 2.4.2.

To reduce the bit-rate of the coder, we should take advantage of the gains of entropy coding, which are evident in Figure 2.9. Note that if better quantizers produce higher entropy signals, we could also explore entropy-constrained quantizer designs. A broader exploration of transform techniques, both for c_2 and the two difference signals, should also lead to bit-rate savings.

Finally, if we are to implement a robust pyramid coder that can operate on speech or general audio, besides designing the parameters for a speech and audio pyramid coder, we need to design a signal classification tool which determines if the input is either speech or general audio. A simple implementation might encode and decode the signal using both speech- and general audio-pyramid coders, and measure which reconstruction has the lowest distortion. The obvious penalty is the added complexity of doing two encodings, decodings, and distortion measurements, so a more advanced speech/audio classification technique

would be desirable. Also, a silence/inactivity detector should be incorporated, especially for speech coding, to efficiently utilize the available bandwidth.

Chapter 3

A Network Model For Simulating Audio Over The Internet

This chapter describes our development and simulation of a network model to evaluate the operation of our priority-transmission scheme under various network conditions. Specifically, we investigate how priority-encoding of layered audio data affects the packet loss within each layer. The goal of our simulations was to generate packet loss traces and statistics for various network congestion schemes and different bit allocations across layered data. This information was then be incorporated into the design process of a layered coder in order to optimize its performance under lossy network conditions. Specifically, our simulation determined the packet-loss rates for the three-layer, priority-encoded bit stream produced by the pyramid coder described in the previous chapter.

Section 3.1 describes the source description we used to model the generation of network traffic, which is based on the model used in [7]. The queuing mechanism and loss discipline of our network model are explained in Section 3.2. In Section 3.3 we discuss the simulations of our source/network model and the data that resulted. Section 3.4 describes methods to improve our model and simulation.

3.1 Source Model

To simulate the source traffic of the network, we chose a system which features a variable total-rate source model. Individual sources are modeled as simple birth-death processes [9], illustrated in Figure 3.1.

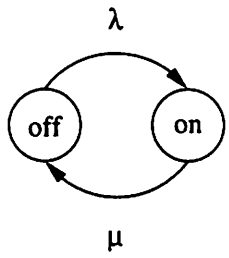

Figure 3.1: Single source on/off model.

In this model, each source has exponentially distributed alternating periods of activity and inactivity with average durations of $1/\mu$ and $1/\lambda$, respectively. The stationary distribution for a single source is

$$P_{\text{on}} = \frac{\lambda}{\lambda + \mu} \tag{3.1}$$

$$P_{\text{off}} = \frac{\mu}{\lambda + \mu}. \tag{3.2}$$

This overall source model was generated by aggregating N independent on/off sources. This leads to the Markov chain shown in Figure 3.2, which describes the total number of active sources. This Markov chain has often been used to model a group of packet voice sources with silence suppression [7, 12]. Note that we implicitly assume that our speech sources utilize silence suppression. To deal with continuous audio, a different model should be used. For example, if studying traffic from a real-time music server, one would expect a group of one-time data streams each of longer duration than the typical alternating exchanges in 2-way voice conversations.

The state variable $n(t)$ describes the total number of active sources at time t. When

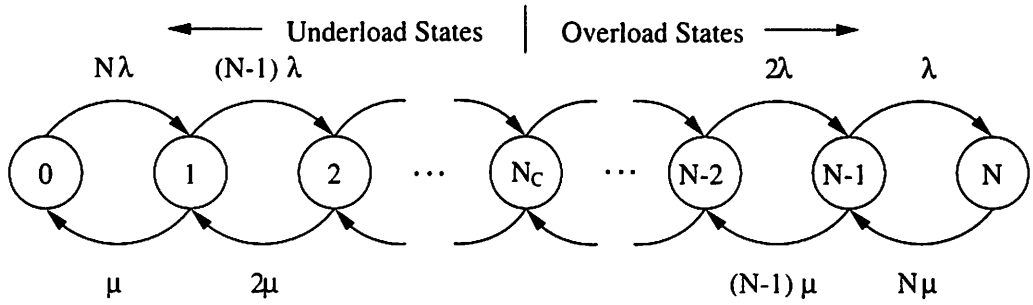

Figure 3.2: Markov chain for N-source model. States are the number of active sources, $n(t)$.

in state n, the transition rate to state $n-1$ is $n\mu$, and the transition rate to state $n+1$ is $(N-n)\lambda$. The stationary distribution of the aggregate source model is binomial since we have N independent and identically distributed on/off processes. Each process has "on" probability $p = P_{\text{on}}$ and "off" probability $q = 1 - p = P_{\text{off}}$, so the stationary probability π_n of having n sources active is given by

$$\pi_n = \binom{N}{n} p^n q^{N-n} = \left(\frac{N!}{n!(N-n)!}\right)\left(\frac{\lambda^n \mu^{N-n}}{(\lambda+\mu)^N}\right), n = 0, \ldots, N. \tag{3.3}$$

The expected number of "on" sources $E[k]$ follows from the binomial distribution of the process and is given by

$$E[k] = Np = \frac{N\lambda}{\lambda+\mu}. \tag{3.4}$$

The capacity C of the channel is defined in terms of N_C, the total number of active sources it can support without loss. The case of interest to us is when N_C is less than the total number of sources N. We define the statistical gain G of the network to be N/N_C. When active, each source produces a total of b bits per input sample so that for an input sample rate of R samples/second (sps), the individual source bit-rate B is $B = Rb$. The channel capacity in terms of bit-rate is $C = N_C B$. States $\{n : n \leq N_C\}$ are underload states with no packet loss while states $\{n : n > N_C\}$ are overload states with packet loss.

Each source uses a two-stage, three-layer pyramid coder similar to the one shown in Figure 2.2. Each source's three layers are packetized individually using fixed-length packets of size L_P bytes, so each source produces three separate data streams. It is assumed that each packet contains an identifier specifying what layer of data, and hence what priority data, it contains. Since the packet size is fixed, the packet rate R_i of Layer i is proportional to the fraction of b allocated to that layer, and can be calculated as

$$R_i = \frac{R \cdot b \cdot f_i \text{ bits/second}}{L_P \text{ bytes/packet}} = \frac{Bf_i}{8L_P} \text{ packets/second.} \tag{3.5}$$

where f_i is the fraction of b allocated to Layer i and is equal to α, β, or γ for $i = 0, 1$, or 2, respectively. We chose a constant-packet-length scheme over a constant-packet-rate scheme because the latter incurs a large packet header overhead for small values of f_i (small values of f_i correspond to small packet sizes). The disadvantage of fixed-length packets is that as f_i decreases, the length of time between transmission of Layer i's packets increases. This leads to increases in the source packetization delay, which is equal to $1/R_i$. For real-time

audio transmission large delay is undesirable, so for a fixed-packet-length scheme L_P and f_i should be chosen so as to not cause unreasonable delays.

Recall from Chapter 2 that a value of 0 for any f_i corresponds to not coding Layer i at all. For example, with $b = 8$, values of $\alpha = 1$ and $\beta = \gamma = 0$ correspond to 8-bit quantization (by Q_0) of the input signal x. A choice of $\beta = 0$ and $\alpha = \gamma = 0.5$ corresponds to 4-bit quantization of d_0, 16-bit quantization of c_2, and the elimination (zeroing out) of d_1 from Figure 2.2 so that $\hat{c}_2$ is directly upsampled and interpolated twice to produce d_0.

3.2 Queuing Discipline and Loss Mechanism

The network is modeled using a single-hop, three-priority queue. The N sources are completely independent, and since their packet production times are determined solely by the random times at which each source turns on, the packets have independent phases across sources. Figure 3.2 is a diagram of the network model, and shows N sources producing three layers of data, all connected to a single node. The priority queue at this node is a FIFO-queue of length L_Q packets which employs a priority drop-tail mechanism. Packets from all three sources arrive at the tail of the queue and are served when they reach its head. When the queue fills up due to congestion, the low-priority packet closest to the tail is dropped. If there are no low-priority packets at all in the queue, the middle-priority packet closest to the tail is dropped, and if there are no middle-priority packets, the high-priority packet at the tail of the queue is dropped. Packets thus retain their order not only within each priority but across priorities as well. This scheme was chosen in contrast to the multiple-queue system explored in [7], which—for a three priority system—employs priority service of three separate queues and a packet timeout loss mechanism.

For purposes of comparison, we also simulated a network system which ignored packet priorities and simply dropped a packet at random when the queue filled up. Our simulation still used the source model described above, so that sources were assumed to be sending three streams of data from a three-layer pyramid encoder. But since the priorities of the streams were ignored, each layer experienced the same loss rate. For such a setup, since we can ignore the individual layers, we can more easily calculate a closed-form expression estimating the expected packet loss rate for a given value of G. Equation 3.4 gives a formula for $E[k]$, the average number of sources active and generating packets. We can also formulate the average number of sources that are on when N_c other sources are already on

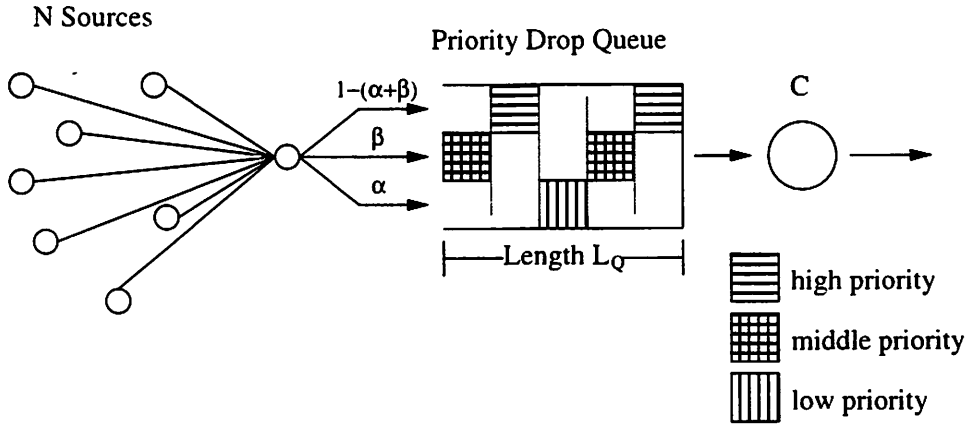

Figure 3.3: Network model illustrating a priority-drop queue and sources producing three layers of data.

as

$$E[N_{ac}] = \sum_{k=N_c+1}^{N} \pi_k(k - N_c). \tag{3.6}$$

If we make the approximation that when N_c+i sources are on, i sources' packets will be lost (this approximation ignores packetization effects and the ability of the queue to temporarily enable transmission of more than N_c sources' packets), we can treat $E[k]$ as the expected number of sources generating packets, and $E[N_{ac}]$ as the expected number of sources above capacity whose packets will be dropped. Using Equations 3.4 and 3.6, we can then estimate the fractional packet loss rate as

$$R_L = \frac{E[N_{ac}]}{E[k]} = \frac{E[N_{ac}]}{NP_{on}}. \tag{3.7}$$

Section 3.3 will show that the above is an excellent approximation.

3.3 Network Simulation and Results

3.3.1 Description of the Simulation

The source and network models described in the preceding sections were combined and simulated using the software tool *ns* [13]. This is an event-driven, packet-level simulator embedded in the Tool Command Language, *Tcl* [16]. To simulate the priority queue, a new *ns* link object was created that implements the priority drop-tail queue behavior described in Section 3.2. For purposes of comparison, a link object employing a random-drop queue

$1/\lambda = 0.6s$	Packet Length $L_P = 48$ bytes/packet
$1/\mu = 0.4s$	Queue Length $L_Q = 50$ packets
$b = 8$ bits/sample	$N_c = 24$ equiv. channels
$R = 8$ Ksps	Simulation Time = 1200 s

Table 3.1: Simulation parameters.

ignoring packet priorities was also created. The N sources were created and controlled individually according to the model of Figure 3.1, rather than controlling the total number of sources as modeled in Figure 3.2. An *ns* script controls the overall simulation and schedules events.

The parameters for the simulation are shown in Table 3.1. Values for λ and μ were chosen for an average voice activity of $\lambda/(\lambda+\mu) = 40\%$. The coder input rate is $R = 8$ Ksps and each source produces $b = 8$ bits/input sample, so the total output rate of each source is $B = 64$ Kbps. This rate is divided among the three layers according to the values of α, β, and γ. Though the simulation can be applied to any source coder producing three layers of data, α, β, and γ were chosen to comply with the pyramid coding constraints of Equations 2.9–2.11. The simulation was run over all possible values of (α, β, γ) satisfying these constraints, which corresponds to 81 possible 3-tuples when $b = 8$.

3.3.2 Simulation Results

The priority-drop network simulation was run for 8 values of G (ranging between 1.5 and 5), each testing all 81 values of (α, β, γ). For purposes of comparison, another set of simulations was run using the same source model and parameters, but with a random-drop queuing mechanism that ignored packet priorities. For a fixed G we expected the random-queue loss rate to be close to constant across layers and independent of the values (α, β, γ), and we ran the simulations to verify Equation 3.7 and to ensure that our simulation was well-behaved. We thus ended up with a total of $2 \times 8 \times 81 = 1296$ simulations. To run these in a reasonable amount of time, the *Xdistribute* [19] program was used to distribute individual *ns* simulations across a large number of workstations.

Table 3.2 compares the theoretical and experimental average packet loss rates for the random-drop case. Theoretical loss rates were calculated using Equation 3.7. Experimental loss rates were calculated using the total number of packets generated and dropped across all values of (α, β, γ). The theoretical estimate compares quite well to the experimental

G	Packet Loss %		Maximum
	Theoretical	Experimental	Deviation
1.5	0.001	0.003	0.004
2.0	0.430	0.641	0.123
2.5	6.295	6.280	0.512
3.0	18.561	17.511	0.898
3.5	29.987	28.637	0.929
4.0	38.671	37.503	1.021
4.5	45.490	44.445	0.738
5.0	50.977	50.000	0.645

Table 3.2: Comparison of theoretical and experimental packet loss rates for a purely random-drop queue. Maximum deviation indicates the largest deviation over the individual (α, β, γ)-combination's loss rates from the average experimental loss rate shown.

results. Also, since the experimental loss rate shown was computed by averaging the results of all of the individual (α, β, γ) experiments, Table 3.2 also shows the maximum deviation of an individual (α, β, γ) loss-rate from the experimental average.

Figure 3.4 shows the loss rates of each layer for $G = 3$ and $G = 5$ for a priority queue. The same techniques for determining the value and effects of γ given in Section 2.5 are valid here as well (note that the viewing angle of Figure 3.4 is the exact opposite of the viewing angle of Figure 2.10). It is clear from Figure 3.4 that Layer 2 enjoys the highest priority and has smaller loss than the other two layers. As the value of α, β, or γ approaches 1, the coding scheme moves to a single-priority source, and hence the loss rate of the corresponding layer approaches the random-drop queue loss rate found in Table 3.2. This is easiest to see in the graph of Layer 2 for $G = 5$, which shows the loss rate approaching approximately 50% as γ approaches 1 (α and β approach 0). By themselves, these graphs indicate how packet loss rates change across each layer as (α, β, γ) changes, but do not indicate an optimal operating point. That point depends on the source coder producing the layered data.

Finally, Figure 3.5 shows how the loss rates across layers vary as a function of G, for a fixed value of (α, β, γ). Note that even though γ was fixed at 0.5, which is a relatively high fraction that corresponds to 16-bit quantization of c_2, Layer 2 experiences no loss for $G \leq 3.5$ and only a maximum of 5% loss at $G = 5$. The graph also shows how as G increases, Layer 0's loss rate increases faster than the other layers' loss rates, until it saturates at 100% loss.

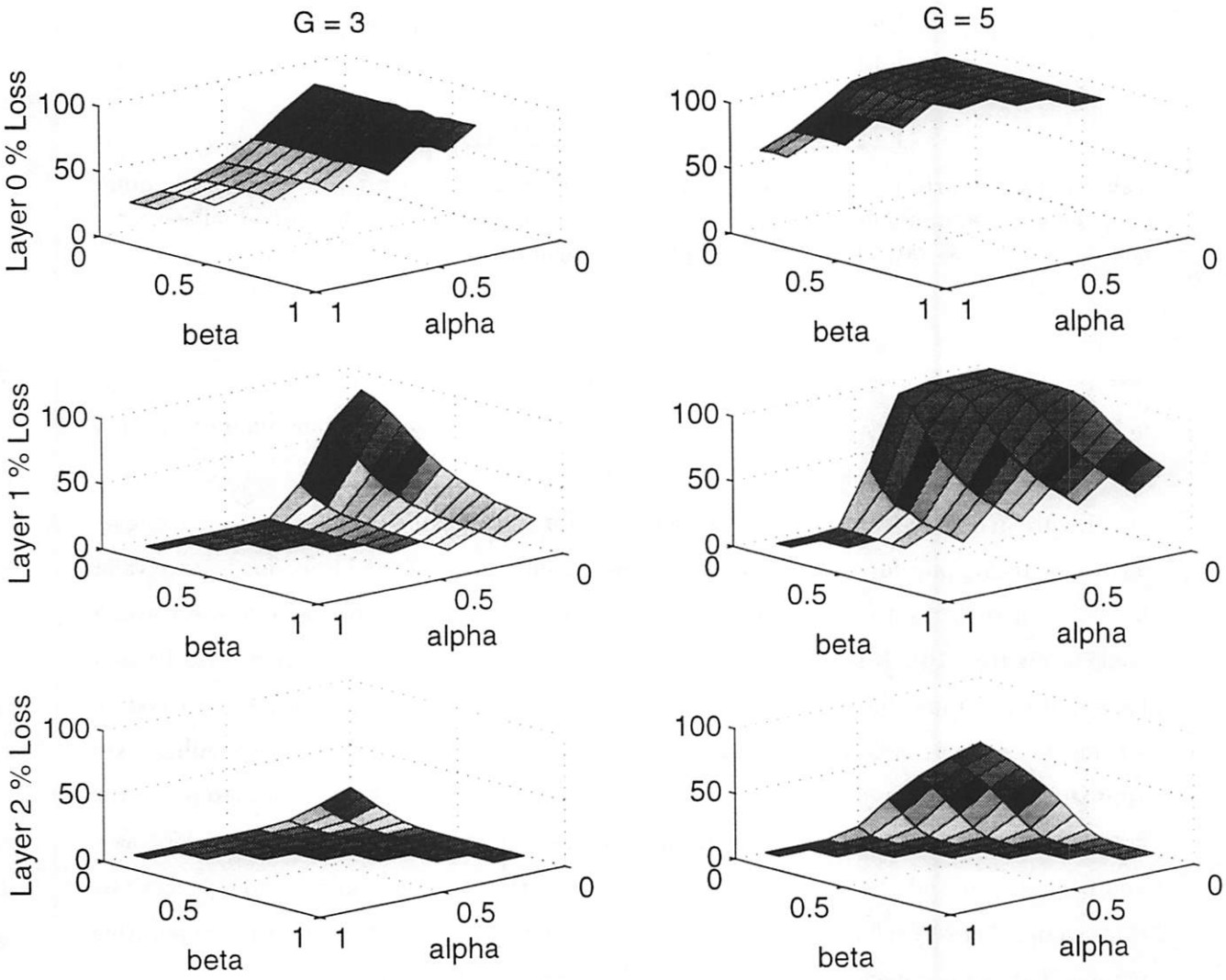

Figure 3.4: Average percent packet loss rates for two values of G when a priority drop-tail queue is used. Loss rates are shown for Layers 0, 1, and 2 as a function of α and β, for $G = 3$ (left column) and $G = 5$ (right column).

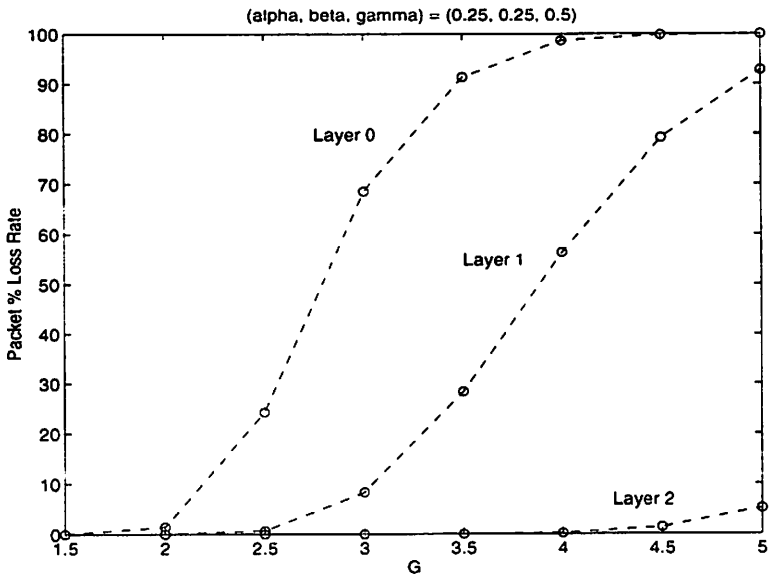

Figure 3.5: Packet percent loss rates across layers as a function of G. Bit-rate allocations (α, β, γ) were fixed at $(0.25, 0.25, 0.5)$.

3.4 Future Work

One aspect to improve in the future is the network traffic model. Currently, the simulation traffic is homogeneous—it all comes from the N source audio model. To better reflect the true nature of the Internet, other types of traffic could be included in the simulation, including email, ftp, and http activity. A model that takes this background traffic into account should be developed so that congestion in our simulation does not arise only from audio sources.

We could also explore the transmission of layered audio over the MBone via different multicast groups. Using the framework for layered multicast transmission described in [15], we can allow receiver-based adaptation to local-network conditions by transmitting the audio layers across different multicast groups. Receivers can then adapt to network conditions by subscribing to as many groups as their local network's capacity allows.

The current Internet is not capable of providing different qualities of service. If we wish to take advantage of the benefits priority transmission can provide, we should explore the application of techniques like Priority Encoding Transmission [1] to the transmission of layered audio over the Internet.

Finally, the impact of f_i and L_P on end-to-end delay should be taken into account.

A minimal value restriction could be placed on f_i so that there is an upper bound on the packetization delay. We could instead allow variable packet lengths L_P across the layers and/or time.

Chapter 4

Joint Source/Channel Coding of Audio

This chapter explores the design of audio coders when the effects of the transmission channel are considered. The channel of interest is a packet switched, priority network. The separation principle states that the source and channel codings can be performed independently, but it holds only for a set of idealized circumstances [5] which are violated in the networks of interest. Therefore, we should be able to improve performance by employing joint source/channel coding design techniques. We have already taken the priority-transmission characteristics of the channel characteristics into account by choosing a layered, redundant-expansion coding scheme. Section 4.1 describes the further refinement of our design through incorporation of the network loss effects. Our goal was to match the bit allocations of our source to the conditions of the network. Section 4.2 discusses the broader issue of determining the optimal redundancy that should be added for error protection over a lossy channel.

4.1 Pyramid Coder Design and Performance for Lossy Networks

4.1.1 Coder Parameters

We tested our pyramid coder's performance over all possible bit allocations when a total of 8 bits were allocated per input sample. We used the network model and parameters

discussed in Section 3.3. The value of the statistical gain G was varied between 1.5 and 5.0 to simulate different levels of network load. We chose the following parameters for our coder: length-19 windowed sinc filters for D and I; automatic-σ scaling uniform quantizers for Q_0, Q_1, and Q_2; no transform coding; and no entropy coding. Because we wanted to explore the performance of different bit layer allocations constrained to a constant total bit rate, we chose not to entropy-code the quantized signals. Length-19 windowed sinc filters were chosen for their superior performance to other designs studied in Section 2.4.2. And though all three uniform quantizer designs we studied in Section 2.4.1 had near equal performance at 64 Kbps (the rate of our test data), we are interested in the coder's performance when packets are lost and the average data rate reaching the receiver is less than 64 Kbps. As such, we chose automatic-σ quantization over the other two schemes because of its superior performance at lower rates.

4.1.2 Computing the Distortion

Our goal in combining our pyramid coder design with network loss information was to determine how a given design performed under lossy conditions. For a given bit budget b and network load represented by G, we want to see how the loss rates across the layers affect the performance of the possible (α, β, γ) bit allocations. We have already seen how the coder performs under lossless network conditions in Section 2.5; now we want to see how (and if) the best choice for (α, β, γ) varies as we introduce loss. A significant difficulty in computing the performance of pyramid coding under loss arises from the signal dependencies inherent in the coder's structure. Due to each layer's strong dependencies on the quantized signals of higher layers, it is very difficult to determine how each individual layer affects the MSE of the reconstruction. Given a specific G and bit allocation choice of (α, β, γ), and the corresponding layer loss percentages (P_0, P_1, P_2), there is no easy way to formulate the expected SNR of the reconstruction. As a result, we determined performance by encoding a set of data, distributing loss probabilities across each layer's packets, and decoding the resulting lossy signal in order to calculate the MSE.

For a set of test data, we used 21.5 seconds of a 16-bit linear, 8000 sps speech signal consisting of 9 different speakers each speaking one sentence. Silence between speakers was removed, since the source activity probability of 40% used in our network model is based on a speech coder with silence suppression. For each value of (α, β, γ), we encoded the data

and packetized each of the signals in fixed length packets of 48-bytes. Then, for each G, we dropped packets with the probability determined by our network simulations. The lossy s_i signals were decoded to calculate $\hat{x}$ and the resulting MSE.

Note that for a fixed bit-rate, our pyramid coder does not have a fixed performance when no data is lost. The fact that Figure 2.10 is not flat is clear evidence of this. As an example of a coder that does have a fixed performance, consider a simple two-layered coding scheme that directly quantizes an input signal and sends the αb most significant bits in high priority packets and the $(1-\alpha)b$ least significant bits in low priority packets. When no packets are lost we get all b bits regardless of the value of α. Only when simulating for network loss would the value of α take on any significance. On the other hand, our pyramid coder has a specific (α, β, γ) combination that results in the best performance under lossless network conditions. As a result, our coder is predisposed to a specific operating point before network loss is taken into account.

4.1.3 Performance Results

Table 4.1 summarizes the performance of our coder as G varies. For each G it shows the best SNR achieved, the values of (α, β, γ) that achieved it, and the corresponding layer loss rates. For all but two values of G, the best choice of (α, β, γ) remains constant at $(0.875, 0.0625, 0.0625)$, corresponding to spending $(7, 1, 2)$ bits on Layers 0, 1, and 2. This result is somewhat counterintuitive because as overall loss rates increased, we expected to see a greater portion of the bit budget allocated to Layers 1 and 2, which have a higher possibility of being transmitted due to their higher priority. This behavior is due to the automatic-σ scaling uniform quantizers.

The data underlying the rate-distortion performance curves of the quantizers shows similar trends for all three types of quantizers. First and foremost, if the rate is increased by increasing b_0 while keeping b_1 and b_2 fixed, the performance always improves. This makes intuitive sense: since the total distortion depends only on the quantization error of Layer 0, increasing the resolution of the quantizer should improve the coder's performance. But it is not always true that when either b_1 or b_2 is increased, and the other two layers' bit allocations are kept constant, performance improves. For example, for many fixed values of b_0 and b_1, the coder using automatic-σ quantizers exhibits the following behavior. As b_2 is increased from 0 (no coding) to 1, there is a positive jump in SNR. This corresponds

G	Best SNR	Corresponding Values			Corresponding Loss %		
		α	β	γ	Layer 0	Layer 1	Layer2
1.5	47.05	0.875	0.062	0.062	0.00	0.00	0.00
2.0	38.22	0.750	0.062	0.188	0.48	0.00	0.00
2.5	26.32	0.625	0.375	0.000	9.30	0.00	NaN
3.0	21.88	0.875	0.062	0.062	21.76	0.00	0.00
3.5	19.67	0.875	0.062	0.062	34.97	0.01	0.00
4.0	18.80	0.875	0.062	0.062	44.92	0.01	0.00
4.5	17.89	0.875	0.062	0.062	52.94	0.01	0.00
5.0	17.23	0.875	0.062	0.062	59.33	0.02	0.00

Table 4.1: Performance as a function G for a pyramid coder using automatic-σ scaling uniform quantizers. For each value of G, the values of α, β, and γ that achieved the best SNR are shown, as well as the corresponding layer packet loss rates for these choices.

to the prediction gain due to coding d_1 as the difference between the prediction based on Layer 2. But as b_2 increases further, performance may increase slightly, but eventually it either levels off or even drops. The same behavior is seen when b_0 and b_2 are fixed and b_1 is increased. The variance of d_1, the unquantized Layer 1 signal, is a function of b_2, the number of bits spent quantizing the coarse signal c_2 upon which d_1 is dependent. Inspection of d_1's variance as a function of b_2 showed that its greatest reduction came as b_1 went from 0 to 1, and only incremental reductions were achieved by further increasing b_1. This indicates that for the automatic-σ quantizers, encoding c_2 or d_0 with a small number of bits produces a significant performance gain over not encoding them at all. We also observed a similar yet slightly different "leveling-off" behavior in the other two types of uniform quantizers. There was often an initial performance drop when going from $b_1 = 0$ to $b_1 = 1$, or from $b_2 = 0$ to $b_2 = 1$. This was due to the very coarse low-bit quantization that tended to increase the power of the next layer's difference signal. After b_1 or b_2 reached 3–4 bits and the quantizer resolution increased, the performance was better than not quantizing the signal at all.

With the above in mind, we can begin to understand the static nature of the optimal (α, β, γ) of Table 4.1. Though priority coding cannot effect any change in the total amount of data lost in a congested network, it can give us control over what data is lost. It might seem that as the loss increases, we should decrease α and thereby increase our allocations to Layers 1 and 2, knowing that a greater percentage of their packets will get through than Layer 0's. However, there is negligible performance gain by increasing b_1 or b_2 above 1 or 2 bits when b_0 is constant. Thus, when operating under a fixed total-rate constraint, there is

even less gain to be had by increasing b_1 or b_2, because doing so would force a decrease in b_0. We therefore do best by fixing with (b_0, b_1, b_2) at $(7, 1, 2)$, the combination that provides the benefits of both predictive coding and accurate quantization of d_0. Increasing b_1 or b_2 beyond these values *would* result in production of more high priority traffic having a better chance of successfully being transmitted, but the resulting decrease in the amount of low-priority traffic would cause an even higher loss rate for Layer 0. Since there is little performance gain to be had by increasing b_1 or b_2 beyond a couple of bits, keeping them minimal and maximizing b_0 is more efficient.

For comparison, the results of maximum-range uniform quantizers are shown in Table 4.2. We see a similar consistency in the optimal (α, β, γ) choice. Under conditions of practically no loss ($G = 1.5$), this coder does best by direct, single-layer quantization of the input. However, as the load increases, more loss ensues and we see benefits of layering the data and getting a high-priority signal consistently through. The recurring choice of $(\alpha, \beta, \gamma) = (0.625, 0.375, 0)$ corresponds to $(b_0, b_1, b_2) = (5, 6, 0)$. The jump from $b_1 = 0$ to $b_1 = 6$ is again indicative of the poor performance that maximum scale uniform quantizers have at low bit levels. After reaching $b_1 = 6$, there is little gain to be had by finer quantization of c_1, so the choice remains the same as G increases. The coarseness of the maximum-range quantizers at small bit allocations is also the likely reason that $b_2 = 0$ for the optimal bit allocation. The minimum number of quantization bits for c_2 needed to improve performance over not coding Layer 2 at all does not appear to be worth the expense of the necessary decrease in b_0 or b_1. Preliminary results with Lloyd-Max quantizers also indicate a similar consistency independent of network load. They also show a more equal distribution of bits for each layer for the optimal (α, β, γ) choice.

The fact that the optimal bit allocation for most network load conditions is identical to the optimal lossless allocation does not imply that the priority-encoding and transmission of the layers is unimportant. Table 4.3 compares the best priority transmission performance from Table 4.1 with what happens when a pyramid encoding scheme is still used but packets are dropped randomly, regardless of priority. It shows that priority transmission gains as much as 17 dB over random-drop.

Finally, in Figure 4.1 we show the performance of a two-layer pyramid coder using automatic-σ scaling quantizers as a function of G. As G increases, the maximum performance is achieved at a smaller value of α. In other words, in this case we do gain by matching the high priority bit allocation to the network loss conditions. However, this is

G	Best SNR	Corresponding Values			Corresponding Loss %		
		α	β	γ	Layer 0	Layer 1	Layer2
1.5	35.66	1.000	0.000	0.000	0.00	NaN	NaN
2.0	28.92	0.875	0.000	0.125	0.33	NaN	0.00
2.5	23.26	0.625	0.375	0.000	9.30	0.00	NaN
3.0	19.27	0.625	0.375	0.000	29.16	0.00	NaN
3.5	17.05	0.625	0.375	0.000	47.54	0.00	NaN
4.0	16.32	0.625	0.375	0.000	62.07	0.00	NaN
4.5	15.38	0.625	0.375	0.000	72.85	0.00	NaN
5.0	14.77	0.625	0.375	0.000	81.48	0.07	NaN

Table 4.2: Performance as a function of G for a pyramid coder using maximum range uniform quantizers.

most closely related to the fact that in the two-layer case, increasing b_1 past a few bits continues to increase the SNR. The same graph for $\gamma = 0.062$ is not shown, for it simply illustrates an increase in performance for all G as α varies from 0 to its maximum possible value of 0.875.

4.2 Redundancy versus Network Congestion

An important issue in the transmission of real-time data over lossy networks is the use of redundancy to protect against channel loss. To study this question, we used a simple protection scheme and tested a wide range of redundancies. In similar fashion to the Robust Audio Tool discussed in Chapter 2, we added protection for each frame to the frame immediately following it. While RAT adds only a fixed-rate amount of data protection coming in the form of a 4.8 Kbps LPC encoding, we instead added the r most significant bits (MSB's) of the current frame's samples to the following frame. This scheme allowed us to try a range of redundancies between the extreme cases of no redundancy ($r = 0$), and 100% redundancy ($r = b$). Note that this MSB scheme is certainly not the optimal redundancy possible for a given percentage. At the very least, we could take advantage of the correlation present in the r MSB's and use vector quantization with entropy coding to reduce the average bit rate. Or we could use lossy compression of the original signal for our redundancy, as RAT does. However, many existing compression schemes either operate at a few rates or have complex implementations. Using them would make it difficult to study a

G	Priority SNR	Equivalent Random SNR	Best Random SNR	Corresponding Values		
				α	β	γ
1.5	47.05	47.05	47.05	0.875	0.062	0.062
2.0	38.22	28.01	34.22	0.750	0.188	0.062
2.5	26.32	12.68	13.29	0.375	0.500	0.125
3.0	21.88	5.32	8.11	1.000	0.000	0.000
3.5	19.67	4.19	5.33	1.000	0.000	0.000
4.0	18.80	3.62	4.31	0.250	0.688	0.062
4.5	17.89	3.36	3.51	1.000	0.000	0.000
5.0	17.23	2.16	3.01	1.000	0.000	0.000

Table 4.3: Pyramid coding comparison of priority-drop vs. random-drop transmission Priority SNR refers to the best priority-drop performance. "Equivalent Random SNR" refers to the performance when the same layer allocations giving the best priority-drop performance are used in a random-drop transmission scheme. The last four columns give the best random-drop performance and the values of α, β, and γ used to achieve them.

wide variety of redundancies. We are thus introducing this MSB scheme as a tool to study the general behavior of adding redundancy for error protection, and not as an optimal error protection scheme for the Internet.

We begin by developing signal and noise models for our proposed system. We treat the input x as a sequence of samples representable by b bits, $x \in (-(2^{b-1}-1), \ldots 0, \ldots 2^{b-1})$. We assume a silence suppression scheme at the source and a uniform distribution p_x, which is a good approximation for companded voice samples during periods of speaker activity [7]. We calculate the source power as:

$$J_s = \sum_x p_x x^2 = \frac{1}{2^b} \sum_{x=-(2^{b-1}-1)}^{2^{b-1}} x^2 = \frac{2^{2b}}{12} + \frac{1}{6}. \tag{4.1}$$

Lost packets correspond to either partial or total losses of data. Total losses occur when two consecutive packets are lost, so that the redundancy information in the second cannot be used to reconstruct the first. Partial losses occur when a packet is lost but its successor arrives successfully. In this case we can reconstruct the lost packet using the redundancy of the r MSB's of the samples stored in the second packet. Replacing the $b-r$ lost LSB's with their mean value of 2^{b-r-1}, the noise energy of the error given a partial loss is

$$J_{pl} = \frac{1}{2^b} \sum_{x=-(2^{b-1}-1)}^{2^{b-1}} (x - \hat{x}) = \frac{2^{2b-r}}{12} + \frac{1}{6}. \tag{4.2}$$

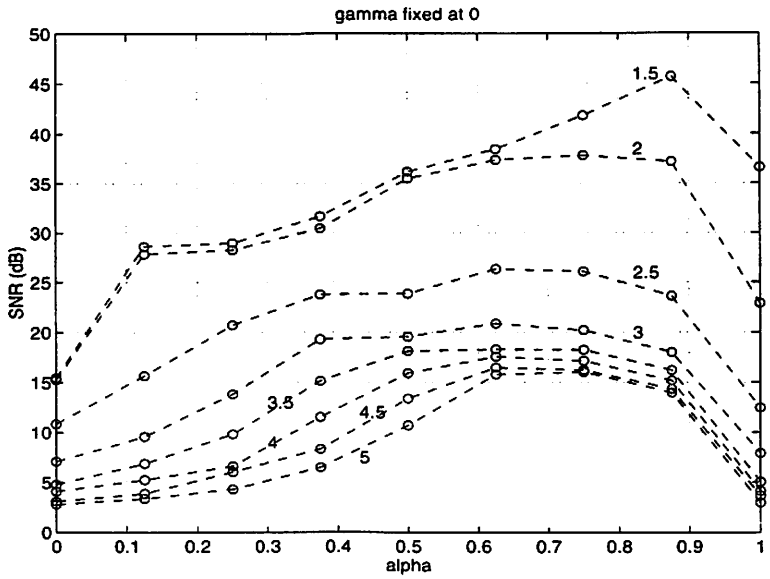

Figure 4.1: Pyramid coder performance as a function of α for fixed $\gamma = 0$ (2-layer coding). Automatic σ-scaling uniform quantizers were used in the coder. Values of G are indicated by the labels on the plot lines.

while the noise energy given a total loss is just J_s, the energy of the signal. The overall expected noise energy is given by a weighted average over these two types of loss:

$$J_n = P_{tl} J_s + P_{pl} J_{pl}. \tag{4.3}$$

where P_{tl} is the total loss probability and P_{pl} is the partial loss probability. The signal-to-noise ratio is

$$SNR = 10 \log_{10}(J_s / J_n). \tag{4.4}$$

To determine P_{tl} and P_{pl}, we ran simulations based upon our existing network model, with the following changes. We only used single-layer sources and random-drop queues since we were not simulating a priority encoding or transmission. Our sources operated on 16-bit audio data at 8000 sps, for a base rate of 128 Kbps. Data was split up into frames of 20 ms, so the base packet size was 320 bytes.

The network capacity was again defined in terms of N_c, the number of these sources that could simultaneously transmit at the *base* rate without loss. For our simulations, we set $N_c = 24$ and looked at the case of a statistical gain of $G = 2.5$. This corresponds to a total of $N = 60$ sources and an average of 24 users actively producing data. We then chose

N_r of these 60 sources to use redundancy such that their packets carried the r MSB's of the samples of the previous packet. These sources transmitted at a rate of $128(1 + r/b)$ Kbps. To calculate P_{tl} and P_{pl}, we kept track of the total number of packets each source employing redundancy sent, how many were dropped, and whether the next packet after a dropped packet was successfully transmitted. We varied r between 0 and b, the extreme values corresponding to no redundancy and 100% redundancy (data repetition), respectively.

Figure 4.2 shows the results of our simulations. Each curve represents the performance for a specific N_r/N percentage of users employing redundancy. Performance is not identical for all curves at 0% redundancy due to statistical variations in our simulations. The results agree with our intuition: when only a small percentage of users add any amount of redundancy to their data, they always do better than they would have without it. However, as the percentage of redundancy-adding users increases, the benefits decrease. This is a clear example of how the best choice for the individual is not the best choice for the group. If the network is in a lossy state and then everyone adds redundancy to their data, the result is even more congestion, greater loss rates, and poorer performance. On the other hand, if the network is hovering around capacity so that packet drops are infrequent, and a single individual adds redundancy to her data, she will do better because her increased data rate has little overall effect on the network. To be fair, we would never expect the Internet to be flooded with 100% of its users adding redundancy to their data. Besides the obvious fact that this would result in poorer performance for everyone, the redundancy is most beneficial for real-time data, for which packet re-transmission is not an option. Though our model is simple and simulates all network sources as audio sources, for intermediate choices of N_r, we can consider the $(N - N_r)$ sources not employing redundancy as representative of other non-real-time traffic sources.

Another important aspect of Figure 4.2 lies in the shape of the individual N_r/N curves. For each curve, there is an initial gain for small amounts of redundancy, but eventually the performance peaks and then declines. At this point, we achieve optimal performance—the best tradeoff between error recovery ability and increased loss rates is achieved.

4.3 Future Work

There is still much that can be done to improve our pyramid coder and produce a practical implementation. The results of our combined source/channel coding experiments

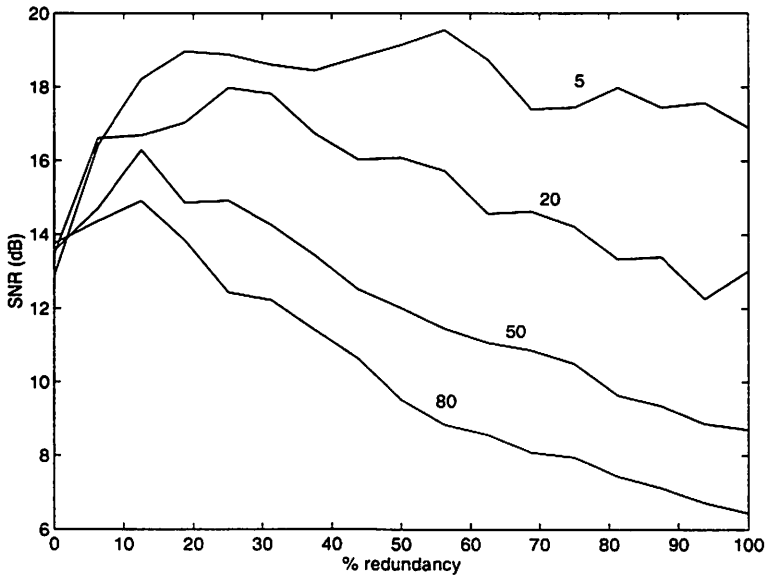

Figure 4.2: Effects of data redundancy as the amount of redundancy used and percentage of total traffic using redundancy varies. The x-axis measures how much redundancy is being added to the base data. The number on each plot line indicates what percentage of users are adding this level of redundancy to their data.

re-emphasize the need for better quantizers and a better distortion measurement. We can also improve the accuracy of our distortion calculations by reflecting the actual times of packet drops into the encoded data. Currently, we use only the total drop percentage to randomly distribute losses. If we instead used the packet-level traces produced by our simulation, we could better assess performance under bursty loss conditions.

Our redundancy experiments point to the existence of an optimal redundancy level for a given network configuration and load level. Though our simulations were for a simple model and redundancy scheme, we believe a more advanced simulation would show the same effects. This remains an open area of research, and we believe its exploration and application to real-time coding designs has the potential to produce meaningful benefits.

Chapter 5

Conclusions

This report has explored the design and performance of a pyramid coding scheme under Internet-like conditions. We studied the advantages a layered coder can exhibit for priority-transmission schemes. After discussing the benefits and limitations of SNR as a performance measurement, we applied it towards the design of the elements of a pyramid coder. We saw how the interdependencies of the coder elements complicated the design process, and how practical considerations force a study of a subset of possibilities. We looked at the performance of a three-layer pyramid coder in terms of both rate-distortion and fixed-rate, variable bit-allocations.

In Section 3.3.2 we saw that priority encoding of a layered signal can yield excellent control over how the network loss is distributed across layers. Even though the current Internet is incapable of providing priority transmission, we discussed how the application of techniques like PET could allow us to emulate priority transmission, and how schemes like RLM can take advantage of a hierarchically layered coder.

When we combined our priority-transmission network simulations with our pyramid coder design, we saw the performance advantages inherent in our layered pyramid coder. We also saw our coder's insensitivity to the network load in terms of the optimal bit allocations. This implies that an intelligent coder design should be able to provide good performance at low loss rates, and a graceful degradation as the loss increases *without* the need for adaptive bit allocation techniques.

Finally, we studied the effects of adding redundancy for packet-error protection. We saw evidence indicating that for a given network condition, there is an optimal amount of redundancy to use, and that adding too much redundancy can actually reduce overall

performance.

There is still much to be done to turn our pyramid coding scheme into an actual Internet audio coder, and we have only scratched the surface of the design process. However, we have obtained a clear idea of where to go and what elements to concentrate on. Pyramid coding produces a naturally hierarchical layering that lends itself to priority transmission and unequal error protection techniques. It has low-complexity and is simple to implement. While it is not the optimal layered audio encoding method, we believe it is worthy of further study and refinement, and has the potential to become a useful approach to Internet audio transmission.

Appendix A

Glossary of Abbreviations and Variables

Abbreviations

ADPCM Adaptive Differential Pulse Code Modulation

bps bits per second

dB decibels

Kbps Kilobits per second

Ksps Kilo-samples per second

LPC Linear Predictive Coding

LSB Least Significant Bits

MBone (Internet) Multicast Backbone

MSB Most Significant Bits

MSE Mean Squared Error

PCM Pulse Code Modulation

QOS Quality of Service

R-D Rate-Distortion

RAT Robust-Audio Tool

RLM Receiver-driven Layered Multicast

SNR Signal-to-Noise Ratio

sps samples per second

vat visual audio tool

Variables

α For a three-layer coder, α is the fraction of the total bit budget spent on the lowest-priority layer. For a pyramid coder, this corresponds to Layer 0.

β For a three-layer coder, α is the fraction of the total bit budget spent on the middle-priority layer. For a pyramid coder, this corresponds to Layer 1.

γ For a three-layer coder, α is the fraction of the total bit budget spent on the highest-priority layer. For a pyramid coder, this corresponds to Layer 2.

λ Rate at which an inactive audio source turns on, expressed in s^{-1}. $1/\lambda$ is the average time length of inactivity.

μ Rate at which an active audio source turns off, expressed in s^{-1}. $1/\mu$ is the average time length of activity.

π_n Stationary probability of having n sources active, $0 \leq n \leq N$.

B Total bit-rate output by the coder of a single audio source.

b Number of bits output by a coder for each input sample.

b_i Number of bits used to quantize Layer i.

C Network capacity expressed in bps.

f_i Fraction of the total bit budget spent on Layer i. For a three-layer pyramid coder, $f_i = \alpha$, β, or γ, for $i = 0$, 1, or 2, respectively.

G Statistical gain. $G = N/N_C$.

L_P Fixed packet length, in bytes, of all packets produced by each of the layers of the pyramid audio coder.

L_Q Length of the priority-drop queue, in terms of number of packets.

Layer i For an J-step pyramid coder, $J + 1$ layers are produced. The $1/2^J$-rate coarse signal is labeled Layer J, and the difference signal of rates $1/2^i$ is labeled Layer i, $i = 0, \ldots, J - 1$.

N Total number of sources in the network.

N_{ac} Number of sources above the capacity of the network that are active. If there are $n(t)$ sources active, $N_{ac} = \max(0, n(t) - N)$.

N_C Network capacity expressed in terms of the number of sources that can transmit without packet loss.

P_{on} Stationary probability of an audio source being on.

P_{off} Stationary probability of an audio source being off.

R The rate of input samples going into the pyramid audio coder.

R_i The packet rate of Layer i, expressed in packets/second.

R_L Packet loss rate expressed as the ratio of total packets lost versus total packets sent.

Bibliography

[1] A. Albanese, J. Blömer, J. Edmonds, and M. Luby. Priority encoding transmission. Technical Report TR-94-039, International Computer Science Institute, Berkeley, CA, August 1994. Available on-line[1].

[2] K. Brandenburg and G. Stoll. ISO-MPEG-1 audio: A generic standard for coding of high-quality digital audio. *J. Audio Engr. Soc.*, 42(10):780–92, October 1994.

[3] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Trans. Comm.*, 31(4):532–540, April 1983.

[4] P. Clarkson. *Optimal and Adaptive Signal Processing.* CRC Press, Boca Raton, FL, 1993.

[5] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[6] ETSI/GSM. GSM full rate transcoding. In *GSM 06.10*, July 1989.

[7] M. Garrett. *Contributions Toward Real-Time Services on Packet Switched Networks.* PhD thesis, Columbia University, New York, NY, 1993.

[8] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Acad. Pub., Boston, MA, 1992.

[9] G. Grimmett and D. Stirzaker. *Probability and Random Processes.* Oxford University Press, Oxford, 2nd edition, 1992.

[10] V. Hardman, M. A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. In *Proc. INET'95*, 1995. Proceedings[2] and paper[3] available on-line.

[1] `ftp://ftp.icsi.berkeley.edu/pub/techreports/1994/tr-94-039.ps.Z`
[2] `http://info.isoc.org/HMP/index.html`
[3] `http://www-mice.cs.ucl.ac.uk/mice/publications/inet95_paper/`

[11] V. Hardman, M. A. Sasse, and A. Watson. Successful voice reconstruction for packet networks using redundancy. Research Note, Dept. of Computer Science, University College of London, April 1995.

[12] S.-Q. Li. A new performance measurement for packet voice transmission in burst and packet switching. *IEEE Trans. Comm.*, 35(10):1083–94, October 1987.

[13] S. McCanne and S. Floyd. *The LBNL Network Simulator.* Lawrence Berkeley Laboratory. Software on-line[4].

[14] S. McCanne and V. Jacobson. *The LBNL Visual Audio Tool.* Lawrence Berkeley Laboratory. Software on-line[5].

[15] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proc. ACM SIGCOMM '96*, pages 26–30, Stanford, CA, August 1996. ACM.

[16] J. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[17] K. K. Paliwal and B. S. Atal. Efficient vector quantization of LPC parameters at 24 bits/frame. *IEEE Trans. Speech Audio Proc.*, 1(1):3–14, January 1993.

[18] D. Pan. An overview of the MPEG/audio compression algorithm. *Proc. SPIE*, 2187:260–73, 1994.

[19] K. Petty and N. McKeown. Xdistribute: A process distribution system. Technical Report M96/67, UC-Berkeley/ERL, November 1996.

[20] S. Quackenbush, T. Barnwell III, and M. Clements. *Objective Measures of Speech Quality.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[21] L. Rabiner and R. Schafer. *Digital Processing of Speech Signals.* Prentice-Hall, Englewood Cliffs, NJ, 1978.

[22] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders. *IEEE Trans. Image Proc.*, 3(5):533–545, September 1994.

[4] `http://www-nrg.ee.lbl.gov/ns/`
[5] `http://www-nrg.ee.lbl.gov/vat/`

[23] T. Tremain. The government standard linear predictive coding algorithm: LPC-10. *Speech Technology*, pages 40–49, April 1982.

[24] J. Tribolet, P. Noll, B. McDermott, and R. Crochiere. A study of complexity and quality of speech waveform coders. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, pages 586–90, Tulsa, OK, April 1978.

[25] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[26] R. Viswanathan and J. Makhoul. Quantization properties of transmission parameters in linear predictive systems. *IEEE Trans. Acoust. Speech Signal Proc.*, 23(3):309–321, June 1975.

[27] R. Warren. *Auditory Perception: A New Synthesis*. Pergamon Press, New York, NY, 1982.