REAL-TIME VOICE TRANSMISSIONS OVER THE INTERNET

BY

DONG LIN

B.E., China University of Science and Technology, 1996

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1999

Urbana, Illinois

ABSTRACT

This research identifies the problems encountered in transmitting voice over the Internet and proposes approaches to solve these problems. The current Internet is not very suitable for transmitting real-time data because its underlying protocols and switches were only engineered to transmit non-real time data. The problems posed by voice over the Internet are studied by conducting a series of experiments. These problems caused by high loss, large delay, and jitter will seriously affect the transmission quality. A good design thus needs to combine different components that solve these problems together. Silence removal and compression is used to reduce bandwidth usage. Jitter buffers are used to smooth the burstiness in the received stream caused by the network. To conceal loss, we investigate existing methods and propose two new nonredundant reconstruction algorithms based on a simple but effective average reconstruction scheme. One is to apply adaptive filtering on top of average reconstruction in order to explore the signal trend and to obtain better estimations. Effects of important filter parameters, such as filter length and adaptation step size, are studied. Another new method, called transformation-based reconstruction method, is developed to handle low reconstruction quality caused by some rapidly changing parts of signals. Its basic idea is to let the sender transform the input voice stream, based on the reconstruction method used at the receiver, before sending the data packets. Consequently, the receiver is able to recover much better from losses of packets than it would without any knowledge of what the signals should be. This method can improve reconstruction quality without significant computational cost and can be easily extended to different interleaving factors and different interpolation-based reconstruction algorithms.

To my parents, sister, and husband

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Dr. Benjamin W. Wah, for all the fruitful and stimulating discussions and the guidance he provided during the course of my graduate study. I have consistently benefited from his professionalism and unwaveringly high standards.

Special thanks go to Xiao Su and Jeff Monks for their valuable discussions. Many thanks also go to past and present members of our research group for all the useful meetings.

I would also like to thank my husband, Zhe Wu, for his encouragement and support.

Finally, I would like to acknowledge the financial support of National Science Foundation Grant MIP 96-32316 and National Aeronautics and Space Administration Grant NAG 1-613.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Voice over the Internet is the process of transmitting voice information—which is traditionally transmitted via public switched telephone network (PSTN)—over the Internet, a packet switched network. The basic steps involve converting analog voice signals to digital format, the compression/translation of the signals into Internet Protocol (IP) packets for transmission over the Internet, and the reverse process at the receiving end.

The integration of voice and data transmissions over the Internet offers an opportunity for significant communication cost savings if reliable, high-quality voice service similar to PSTN [1] can be achieved. Although the benefits of real-time voice transmissions over the Internet are obvious, and there have already been many commercial products for Internet telephony in the market, only a small fraction of users who have tried these applications are willing to adopt and actively use the technology [2]. The most important reason is that the level of speech quality (such as end-to-end delay, jitter, clarity, and continuity), achieved is not comparable to the quality of PSTN [3]. These considerations motivate us to investigate the problem for transmitting real-time voice over the Internet.

1.2 Problem Statement

The quality of most current Internet real-time voice transmission systems is not satisfactory because of the current Internet's delivery and scheduling mechanisms. The Internet has been traditionally designed to support text-based non-real-time data communications, but not real-time voice transmissions, such as Internet phone. These real-time applications have quite different characteristics as outlined here.

The first significant characteristic of real-time applications is their high delay sensitivity. Given strict end-to-end delay and interframe delay requirements for real-time transmissions, packets delayed over a certain time limit are considered lost [4] and cannot be retransmitted by the sender [5]. The current Internet does not support real-time transmissions because it has no special delivery mechanism to differentiate between real-time data and non-real-time data. Hence, all real-time data frames are treated the same way as non-real-time frames and will be dropped or delayed with equal chance under heavy load and congestion. The current Internet also may have large delay variations and loss. Measurements carried out by others [6] and us have shown that the loss rate of packets to some destinations can be as high as 50%.

The second significant characteristic is that most real-time applications do not require data to be 100% precise, unlike services provided by Transmission Control Protocol (TCP), which ensure that all data packets are sent correctly and reliably all the time. This characteristic is very useful because the receiver can tolerate a certain level of loss or distortion of data without significant degradation in performance [7].

The above two characteristics define the potential problems that should be considered in order to develop a high-quality real-time voice transmission system. Reliability and predictability are the two major problems [8]. Reliability ensures the reliable delivery of voice packets so that loss is concealed from users, whereas predictability avoids the delivery of voice packets with excessive delay and to maintain a certain playback rate. These requirements can be measured by quality-of-service (QoS) measures. The key QoS measurements include end-to-end delay, jitter, and loss. End-to-end delays and jitters are key measurements of predictability, whereas loss is the key measurement of reliability. Obviously, reliability is a more urgent problem to be handled because without reliability, predictability is hard to achieve.

The purpose of this thesis is to examine the necessary components of a real-time voice transmission system by studying Internet voice-traffic behaviors and by designing new reconstruction methods to conceal loss and improve transmission quality.

1.3 Related Work

In the current literature, there is no good way to reduce end-to-end delay except by keeping the processing and buffering delay at both ends low, because network delay is not controllable [9]. The use of jitter buffers [10] is universally accepted for controlling jitters, although they will introduce extra delay.

For concealing loss, algorithms can be classified into two major categories: receiver based, and sender and receiver based.

The first class of nonredundant reconstruction methods for concealing loss involve the receiver only. In these methods, only one copy of each voice packet is sent, and the receiver is responsible to recover the lost packets. A common strategy to compensate the loss of a packet is to replay the last packet received during the interval when the lost packet is supposed to be played back. If the length of a bursty loss is short, then this scheme can give reasonable playback quality. Another strategy is to replace the lost packets using a segment of silence or a segment of white noise. These two simple strategies can fill the gap between noncontiguous speech frames received at the receiver and work well when the occurrence of lost frames is infrequent and the frame size is small. However, they do not work well when the length of a bursty loss is large [11]. A third strategy reconstructs the missing data based on data received already.

The second class of reconstruction methods for concealing loss involves both the sender and the receiver. The sender first processes the input data streams in such a way that the receiver can reconstruct the missing data better. Based on the different ways of processing the input data, these schemes can further be split into two subclasses: one that adds redundant control information and one that does not.

There are several methods for the sender to add redundancy to data streams. The first method [12] sends redundant information at the expense of increased bandwidth. Its general idea is to replicate and send the ith packet along with the $(i+1)$st packet so that when a packet is lost, the receiver still has another packet as a backup. Obviously, the network bandwidth is doubled, leading to higher delays and congestions. Another method used is based on forward error correction (FEC) [13, 14] that protects every n packets by inserting a redundant packet containing an error correcting code. This approach increases the bandwidth by $\frac{1}{n}$ and may increase the latency by n times, since in the worst case, n packets have to be received before the lost packet can be reconstructed. A third method is adopted in the MICE project [11] that utilizes a kind of sequence loss protection. The basic idea is to add extra data in the ith packet that contains redundant information for the $(i-1)$st or $(i-2)$nd packet. Although this process reduces the latency as compared to the FEC method, it still requires more network bandwidth.

There are also quite a few algorithms that do not add any redundancy to the data sent. Instead, they utilize the inherent redundancy of the source input stream. A typical method based

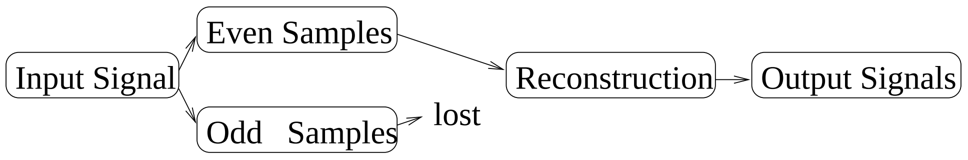


Figure 1.1: Reconstruction in two-way interleaving.

on interleaving transmits interleaved voice samples in one packet and reconstructs approximately lost samples using their surviving neighbors contained in other packets. One simple but effective method [15] is to group all the odd samples into one packet and all the even samples into another, and then send them independently. We call the two packets with the corresponding even and odd samples *an interleaving pair*. When one of the packets is lost, the receiver can easily use the average of samples in the other packet to reconstruct the lost samples (see Figure 1.1). Simple averaging works well for voice signals because most samples are related to their neighboring samples. It is easy, it is fast, and it does not require redundant information to be sent. However, simple averaging may fail when signals are rapidly changing or when signals are totally independent. In the latter case, averaging amounts to adding noise to fill the missing gaps.

In short, a combined sender and receiver reconstruction technique generally improves reconstruction quality significantly over a receiver-only reconstruction technique. Trade-offs must be made to balance between the amount of redundancy and the amount of increased network traffic. Nonredundant schemes based on interleaving and averaging are attractive because they perform reasonably well.

1.4 Our Approaches to Conceal Loss

To overcome the shortcomings of reconstruction using simple averaging, we investigate in this research two new approaches to reconstruct lost packets built on top of interleaving and simple averaging. The performance measure for reconstruction quality is the signal-to-noise ratio (SNR):

$$SNR \;=\; 10\;log_{10}\frac{\sum(s^2)}{\sum(s-\hat{s})^2}$$

where s is the original signal and $\hat{s}$ is the reconstructed signal of s.

One shortcoming of reconstruction using averaging is that it does not track changes in the original signals. To address this problem, the first approach applies adaptive filtering after reconstruction using averaging at the receiver in order to track the voice signals and to give better reconstruction quality. The algorithm works as follows.

The sender

- interleaves the original voice stream into two substreams,
- packetizes the substreams separately, and
- sends the packets to the receiver.

At the receiver, there are two cases to be considered. First, when the receiver receives both packets in an interleaving pair, the receiver

- de-interleaves,
- trains the adaptive filter by assuming one packet is lost, and
- feeds the de-interleaved packets to the sound card.

Second, when the receiver receives only one packets in an interleaving pair, the receiver

- reconstructs using average reconstruction,
- passes the reconstructed stream through the adaptive filter, and
- feeds the output of the adaptive filter to the sound card.

Because adaptive filtering needs time to adapt, it may not perform well when the signals are changing rapidly. This motivates us to develop a new transformation-based reconstruction algorithm that transforms the original signals at the sender before sending them out. The transformations are done according to the reconstruction method used at the receiver in order to enable better reconstruction quality. The algorithm works as follows.

The sender

- transforms the voice stream according to the reconstruction method used at the receiver,
- interleaves the transformed stream,
- packetizes each substream, and
- sends the packets to the receiver.

There are two possibilities at the receiver. First, the receiver receives both packets in an interleaving pair. The receiver then

- de-interleaves the stream, and
- feeds the de-interleaved packets to the sound card.

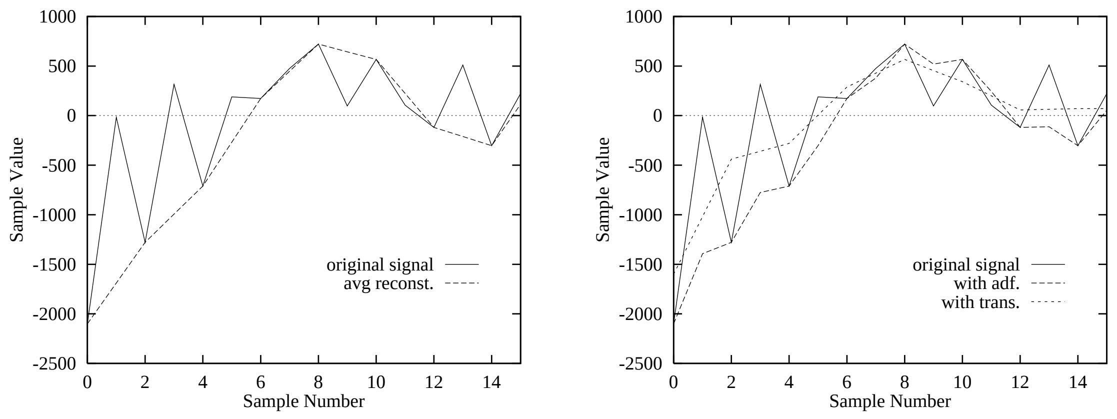
Second, the receiver receives only one packet in an interleaving pair. The receiver then

- reconstructs using simple averaging, and
- feeds the reconstructed stream to the sound card.

To illustrate the basic ideas of the two reconstruction methods, consider a simple two-way interleaved data stream based on a typical segment of voice data with 16 samples. Assuming that the odd samples were lost at the receiver and that the eight even samples were used to reconstruct the missing samples, Figure 1.2(a) plots the reconstructed stream in which a missing odd sample is computed as the average of its two adjacent even samples. In contrast, Figure 1.2(b) shows the reconstructed streams of our proposed reconstruction methods. In the method based on averaging and adaptive filtering, the missing samples are first reconstructed using averaging, and the stream is then passed through an adaptive filter. After adaptive filtering, the reconstructed stream is a better approximation of the original one because adaptive filtering can track the shape of a waveform. However, adaptive filtering is not able to adapt to rapid changes that span only a few samples in the original signals, as shown by samples 1-3. In contrast, the dotted line in Figure 1.2(b) shows the data reconstructed by our transformation-based method. The even samples were first transformed at the sender, and the receiver uses the transformed even samples to reconstruct the missing odd samples. It is obvious that the reconstructed stream based on the transformed samples gives a better approximation to the original stream. As shown by samples 1-3, transformation smooths out rapid changes in the original waveform, while at the same time trying to preserve its trend. Therefore, the reconstructed samples based on averaging are more accurate.

1.5 Protocol Considerations

An ideal protocol for voice transmission should have certain features, such as resource reservation and service-quality guarantees. However, current Internet protocols are far from ideal. Although the Resource ReServation Protocol (RSVP) [16] is standardized as the future protocol for applications



(a) (b)

Figure 1.2: Comparison of reconstruction quality among simple averaging, averaging with adaptive filtering, and averaging based on transformation, assuming that only the even samples are available. The original voice samples are in solid lines. (a) The odd samples were reconstructed by taking the average of its two adjacent even samples (reconstruction by averaging) leading to SNR of 1.69 dB. (b) The data that were reconstructed using averaging and adaptive filtering are in dashed lines with SNR of 3.18 dB. The performance given by the dotted line was obtained by first transforming the even samples before they were sent and by reconstructing the odd samples using simple averaging at the receiver (SNR of 4.03 dB).

requiring guaranteed bandwidth for high-quality transmission of real-time audio and video data, it may not be practical for use in the near future [8]. The most widely supported network-layer protocol, IP, does not provide any QoS guarantees, but only provides connectionless best-effort packet delivery service.

Because network-layer protocols and the underlying switches support only best-effort service, it is not possible to use transport-layer protocols to support real-time transmissions. TCP and User Datagram Protocol (UDP) are the transport-layer protocols for the Internet. TCP is connection-oriented and is responsible for the correct delivery of data by detecting lost data and by triggering retransmissions. Although TCP is reliable, it will result in unacceptable delays when transmitting real-time data. On the other hand, UDP is connectionless, with no guarantee of delivering packets to its destination. With no connection setup at the beginning, every packet may involve additional overhead to be routed, and packets arriving at the destination may be out of order. Hence, UDP requires the application to reorder packets and take care of loss. Although UDP does not provide the underlying support for QoS, it is not necessarily bad because it gives an application more flexibility in addressing application-specific requirements.

Most of the current real-time applications over the Internet are based on UDP and follow the standard Real-time Transport Protocol (RTP) [17]. RTP is not a transport-layer protocol but rather an application-layer protocol. RTP was designed to be embedded in an application that uses an underlying transport-layer protocol. It provides a mechanism to time-stamp packets so that random delays resulting from other network traffic can be compensated by the use of buffers at a destination location. That is, by using time stamps, packets can be buffered and removed from the buffers in a correct sequence. Although it is called the real-time transport protocol, RTP does not contain any mechanism that guarantees the timely delivery of data, nor does it provide any other QoS guarantees. Still, network-layer protocols are the basis for guaranteeing any QoS.

For our system model, we choose UDP as our transport-layer protocol based on the above discussion. Also, our system needs to handle packet loss, out-of-order packets, and delayed packets in order to maintain an acceptable quality.

1.6 Significance of Work

In this research, we carry out a comprehensive study of Internet traffic behavior and develop a generic system model based on the traffic study. Tradeoffs among end-to-end delay, jitter, loss, and bandwidth are considered throughout the design of our system model and the design of our new reconstruction methods. We further propose a new transformation-based reconstruction method to conceal loss and to minimize reconstruction errors.

1.7 Outline of This Thesis

In this thesis, we propose a system model for real-time voice transmissions and focus on solving the packet-loss problem by interleaving and reconstruction.

Chapter 2 investigates the traffic pattern for real-time voice transmissions and proposes the necessary parts the system should comprise. Chapter 3 describes the different components of the system, their approaches, and their main functionalities. Chapter 4 presents two new reconstruction methods to overcome packet losses. The first one combines adaptive filtering and average reconstruction to improve the reconstruction quality. The second one addresses the problem at the sender site, which first transforms the voice stream so that average reconstruction at the receiver will achieve the best results when loss happens. Chapter 5 summarizes the work of this thesis.

CHAPTER 2

INTERNET TRAFFIC EXPERIMENTS

This chapter presents a series of Internet traffic experiments. These experiments were carried out to reveal the underlying loss and delay patterns for voice traffic over the Internet. The results are analyzed, and their implications on voice transmissions are discussed.

2.1 Objectives

The unreliable nature of the Internet is well known. Packets transmitted over the Internet can be delayed, duplicated, or lost. Reliable transmissions of textual data are achieved through retransmissions provided by TCP. The very basic idea of TCP is timeout retransmission under the control of a window-based strategy. The flow-control and retransmission scheme in TCP was designed to work for all network situations with different levels of reliability. Data transmitted are 100% accurate despite delay and loss. The requirements of total correctness but flexible delay tolerance make it possible to have a conservative, universal error correction and handling scheme.

The delivery of voice signals over the Internet, on the other hand, is much more flexible in terms of precision, but more restrictive in terms of delay. The fact that voice delivery can tolerate some loss opens up a set of error handling schemes, including retransmission, forward error correction (FEC), and reconstruction. These schemes will be discussed in more details in Chapter 4. Understanding the loss and delay behavior of Internet traffic is essential in designing an efficient and high-quality voice transmission scheme.

This chapter presents a series of traffic experiments to identify loss, delay, and jitter behaviors, and their relationships. The goal of these experiments is to provide useful guidelines for designing our system model. The experiments are targeted at identifying the following:

Table 2.1: Hosts involved in the Internet traffic experiment.

Host Name	IP Address	Location
`trace4.crhc.uiuc.edu`	130.126.142.44	University of Illinois(UIUC)
`cs.stanford.edu`	171.64.64.64	Stanford University(Western US)
`bart.cc.utexas.edu`	128.83.40.7	University of Texas(Southern US)
`math.mit.edu`	18.87.0.8	MIT(Eastern US)
`public.guangzhou.gd.cn`	202.96.128.111	China
`iasia1.iasi.rm.cnr.it`	150.146.5.13	Italy
`faraday.ee.uec.ac.jp`	130.153.149.28	Japan

- *Characteristics of packet loss* that include the amount of packet loss and its bursty behavior
- *Relationship between loss rate and sending rate* and whether rate adaptation is useful
- *Characteristics of delay* and their effects on the quality of voice transmission
- *Characteristics of jitter* and their relationship to the amount of jitter buffers

2.2 Experimental Setup

In order to draw unbiased conclusions about traffic behaviors over the Internet, we chose sites in our experiments using several US connections as well as a few transcontinental connections. Table 2.1 lists the hosts involved in the experiments. The Stanford University connection, the University of Texas connection, and the MIT connection represent typical Internet connections within the United States. The Guangzhou connection represents a typical US-Chinese connection. The Italian connection represents a typical US-European connection. The Farady connection represents a typical US-Japanese connection.

The experiments were conducted between a UIUC machine and the other machines. Figure 2.1 shows the configuration. During the experiments, the UIUC machine periodically sent 2000 probe packets every hour, each with a sequence number, to the echo ports of the other computers. Meanwhile, the sending time of each packet was saved to disk for future analysis. After the packets were bounced back by the echo service on the remote hosts, the UIUC computer recorded the arrival time of each packet and its sequence number. The statistics were collected once an hour. After

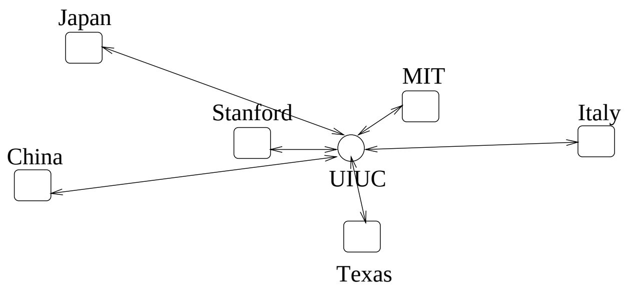


Figure 2.1: Experimental configuration to test traffic behaviors over the Internet.

an hour's experiment, the loss, burst length, round-trip time, and jitter were calculated from the sending times, arrival times, and sequence numbers received.

General network behaviors over a wide range of sending rates and packet sizes were investigated. First, the largest sending rate is bounded by the scheduling ability of the underlying operating system. In our case, the underlying operating system is a Solaris 2.6 that supports a system timer at the precision of tens of milliseconds. Under this restriction, the sending rates tested were varied from 1 to 100 packets per second. Second, the packet size is bounded by the ability of echo service. Most echo services support packet sizes less than 512 bytes, which seems at a first glance to be a big restriction in our experiments. However, with a compression ratio of 8, each sample takes 2 bits (originally 16 bits); therefore, 512 compressed bytes is equal to 2048 bytes (or 250 ms) of uncompressed samples. Because 250 ms of delay is actually too large when end-to-end delay is concerned, the sizes of packets tested range between 5 and 512 bytes.

2.3 Experimental Results and Analysis

2.3.1 Overall loss behavior

To give a general idea of loss behaviors in the Internet, we plot in Figure 2.2 the loss percentages to six destinations during a day for a sending rate of around 17 packets per second. (Detailed behaviors for different sending rates and packet sizes will be discussed in Section 2.3.3.) A first impression of the results is that packet losses are common. All experiments exhibited packet losses, and packet losses varied from 1% to 60%, largely depending on the geographical region where the echo server is located and the time of the experiments.

The connections inside US normally exhibit small amount of losses (below 10%). If only the loss percentage is concerned, voice transmissions inside the US should be practical. However,

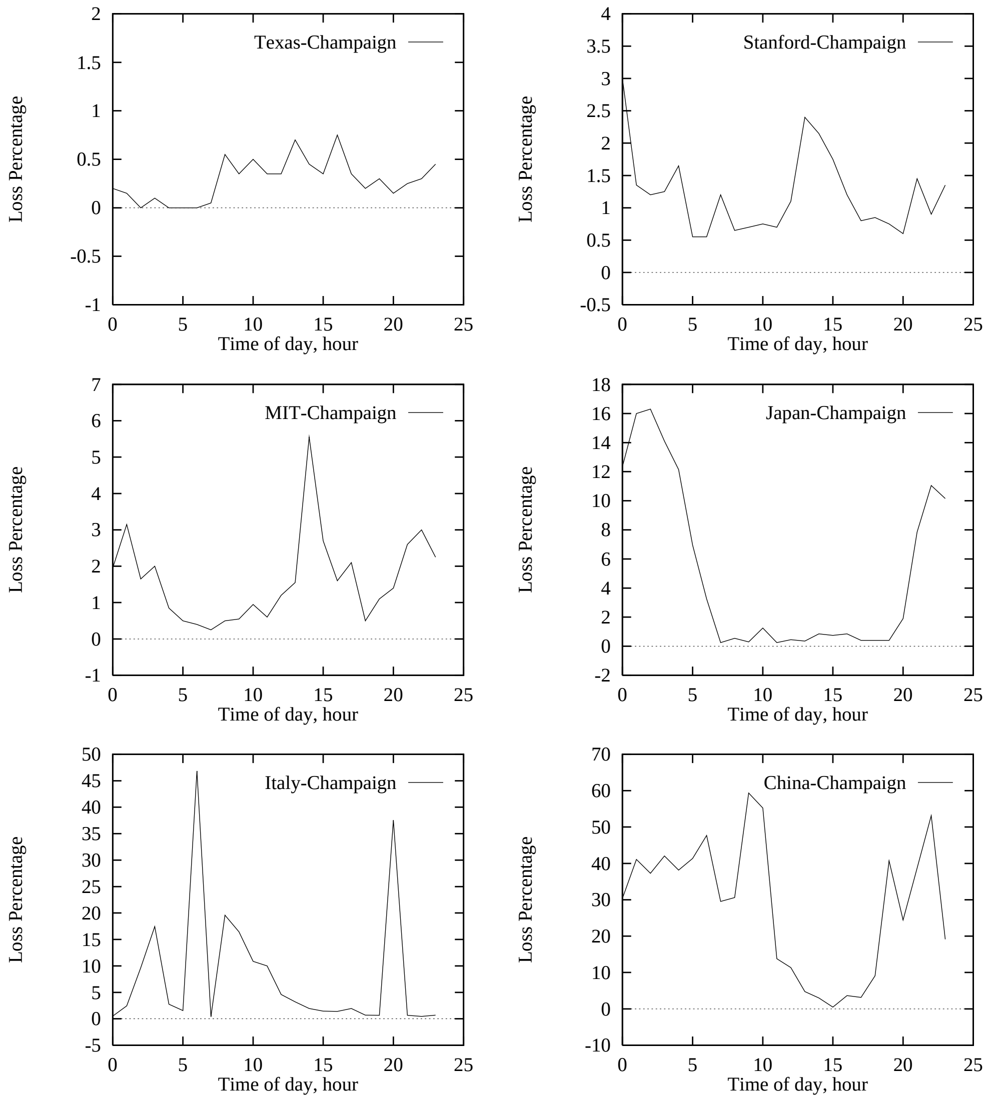


Figure 2.2: Loss percentages to different destinations over a 24-hour period.

the situation changes with transcontinental connections. The loss percentage for transcontinental connections goes up immensely. For example, the connection between the US and China has lost almost half of its total packets. This high loss rate will be a big challenge for real-time system designers.

The results also reveal the relationship between the loss percentage and the time of day. Generally, during working hours, losses are high. The most obvious site is the Japanese one. During working hours in Japan, the average loss is around 8%, whereas after working hours, there is almost no loss. The behavior of the three sites in the US is not as obvious as that of the Japanese site, but still shows more losses during working hours. The Italian site is a little different, besides an obvious, high-loss period during working hours (around 2 a.m. - 11 a.m. Champaign time), there was another peak at its midnight (around 8 p.m. Champaign time). One possible explanation is that it is a File Transfer Protocol (FTP) server and has a high demand at that time.

In summary, loss percentage is connection-dependent and time-varying. This requires the transmission system to have the ability to adapt to different loss behaviors. In order to do this, the system needs to know the current network behavior and to incorporate statistics collected at run-time to adjust its operations.

2.3.2 Length of consecutive packet loss

Figure 2.3 plots the probability distribution functions (PDFs) of consecutive packet losses. For all the six destinations, bursty-loss lengths of 1 and 2 are predominant. These are more clearly shown in Figure 2.4, which plots the probability distribution functions of different burst lengths for a typical hour (11 a.m. Champaign time) with a sending rate of 100 packets per second. As an example, for the Chinese site, bursty losses of lengths 1 and 2 take about 85% of the total number of losses.

The results of testing the correlation between adjacent burst lengths are shown in Figure 2.5 and Table 2.2, respectively. Here, the correlation among adjacent burst lengths is shown both graphically and quantatively for the China-Champaign connection during its working hour, which has the highest loss percentage among the six connections. From Figure 2.5 and Table 2.2, we can see burst length of 1 or 2 are highly correlated. That is, when the current burst length is 1 or 2, the next burst length is most likely to be 1 or 2. Moreover, losses with burst length longer than 3 happen very infrequently (less than 12%). Although long bursts may happen, especially when the sending rate is high (see Figure 2.5), one long burst usually does not imply that the next burst

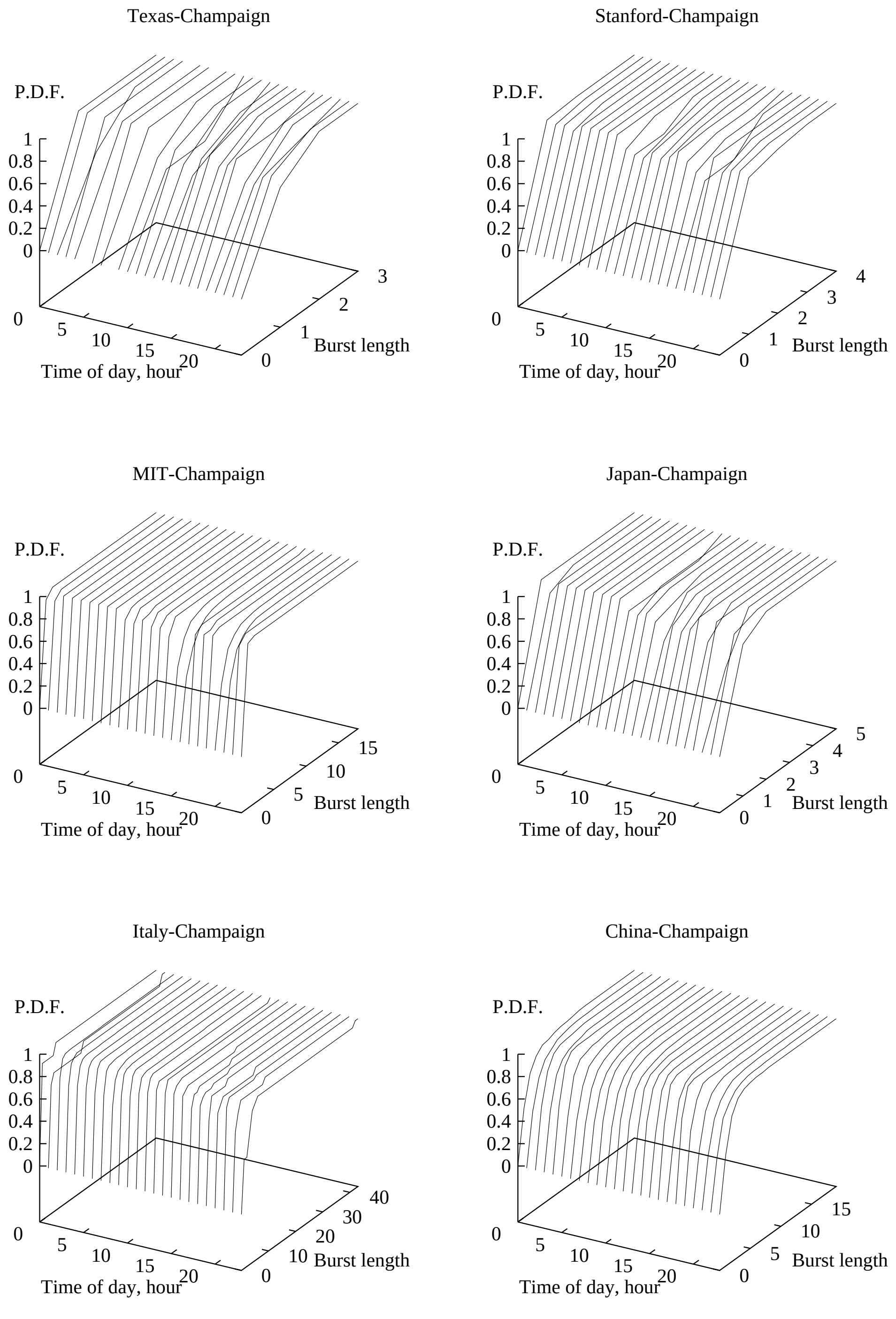


Figure 2.3: Probability distribution function (PDF) of consecutive packet losses.

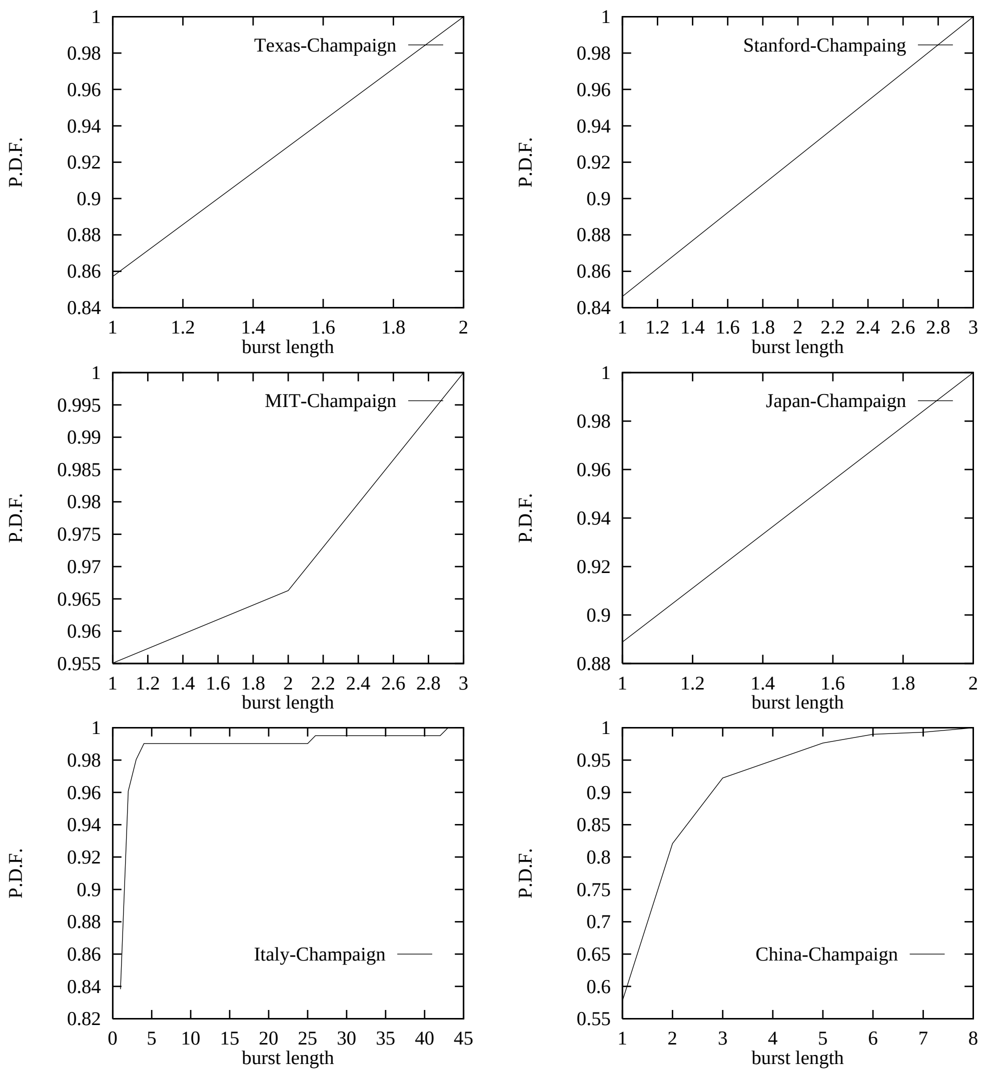


Figure 2.4: Probability distribution function (PDF) of consecutive packet losses to different destinations at a typical hour (11 a.m. Champaign time) with a sending rate of 100 packets per second.

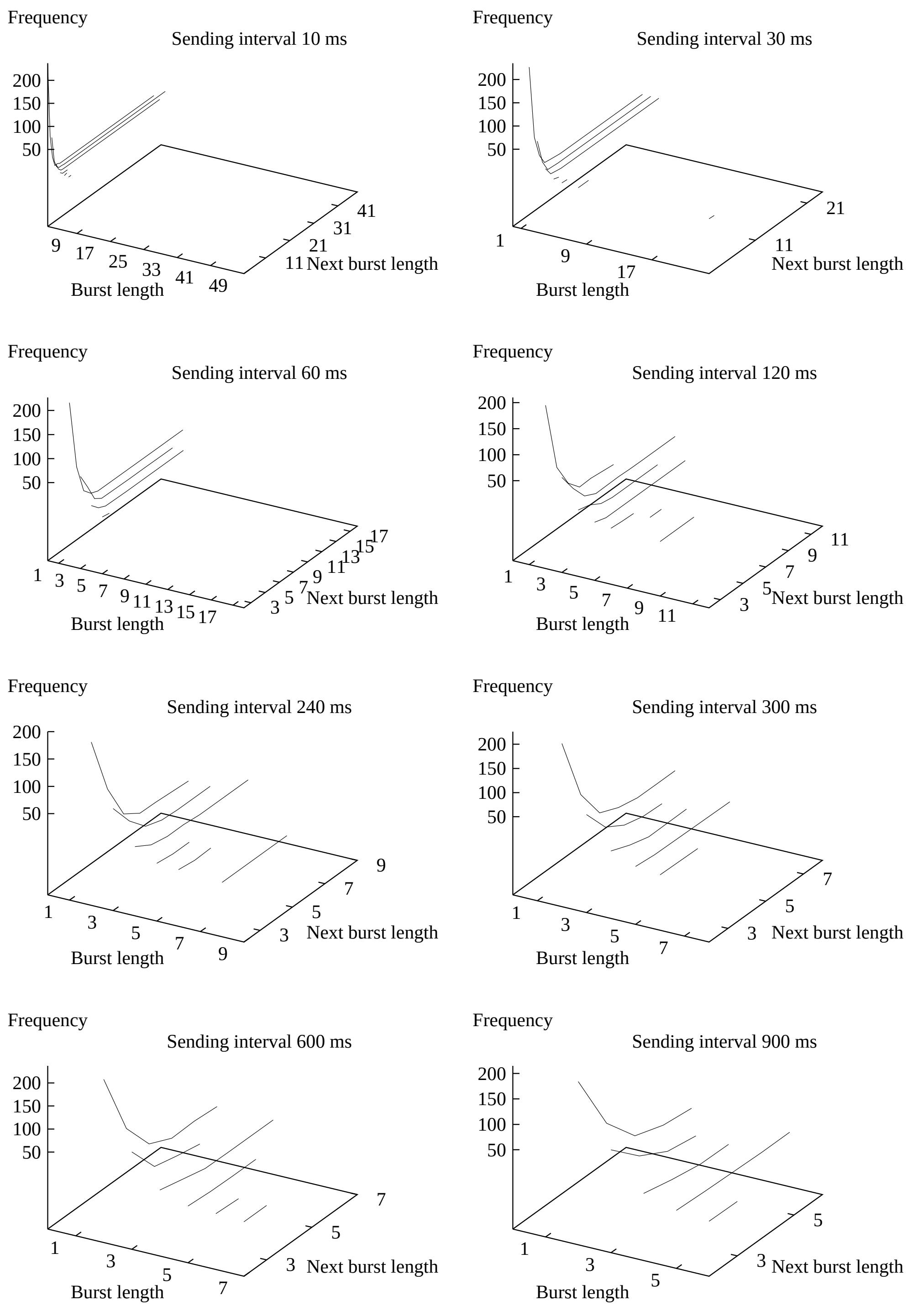


Figure 2.5: Correlation of burst length for the China-Champaign connection with different sending rates.

Table 2.2: An example of the correlation of burst lengths. (2000 packets were sent from Champaign to China with a sending rate of 100 packets per second and 500 bytes per packet.)

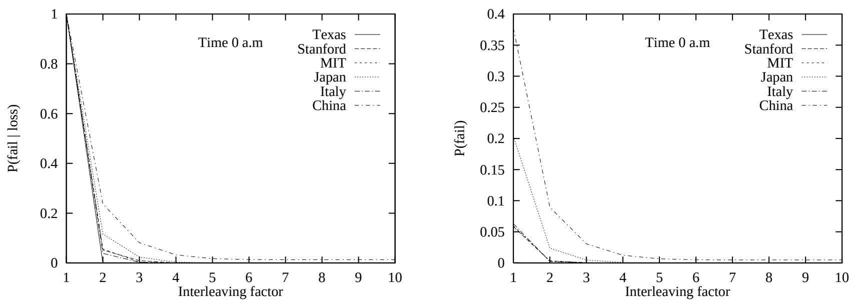
	Conditional probability of next burst length											
Burst length	1	2	3	4	5	6	7	12	43	44	45	48
1	0.68	0.214	0.074	0.010	0.01	0.003	0.003		0.003		0.003	
2	0.68	0.200	0.060	0.020	0.02	0.010						0.01
3	0.63	0.230	0.050	0.030				0.03		0.03		
4	0.75	0.125		0.125								
5	0.33	0.670										
6	0.50	0.500										
12	1.00											
43		1.000										
44	1.00											
45		1.000										
48	1.00											

will also be long.

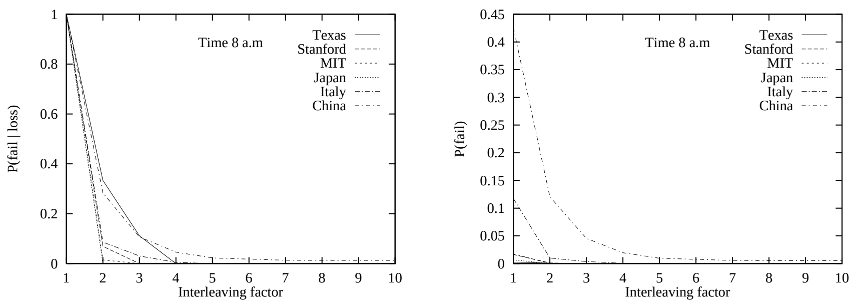
Knowing that the burst length is usually small is very useful for designing a scheme to alleviate the effect of losses. A good method is interleaving that distributes correlated information to different packets. If the burst length is less than the interleaving factor, there are always parts of information received that can be used to recover the lost parts. With an interleaving factor of 2, a bursty loss of length 1 and a bursty loss of 2 with samples belonging to different interleaving pairs can be overcome. With an interleaving factor of 4, a bursty loss of length less than or equal to 3 and a burst length of 4 with lost packets belonging to different interleaving sets can be recovered. In general, with an interleaving factor of i, a bursty loss of length less than or equal to $i-1$ and a bursty loss of length i with probability $(i-1)/i$ can be overcome, whereas bursty losses of length longer than i cannot be overcome.

Figure 2.6 shows the probabilities of bursty losses that cannot be recovered as a function of interleaving factor i. $P(fail \mid loss)$ in the graphs on the left represents the conditional probability that a packet cannot be recovered, given that the packet is in a burst of lost packets. $P(fail)$ in the graphs on the right represents the probability that a packet cannot be recovered in the stream received for a given interleaving factor. Assume that the total number of packets sent is NP, and that bursty losses of length k happen N_k times. The number of packets lost is NL:

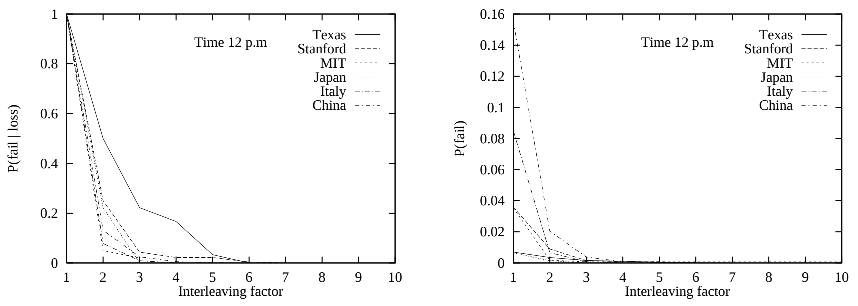
$$NL = \sum_{j=1}^{\infty} j N_j \tag{2.1}$$



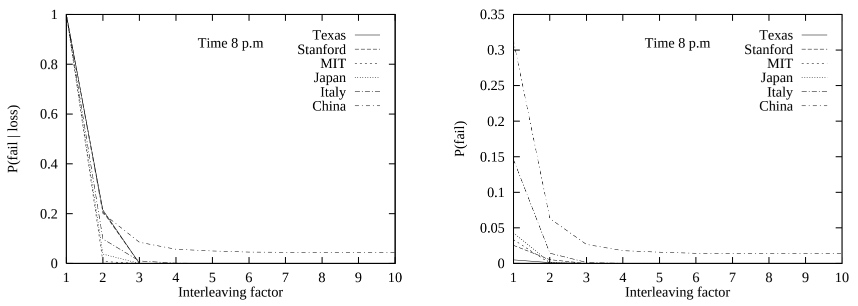
(a) Champaign time 0 a.m.



(b) Champaign time 8 a.m.



(c) Champaign time 12 p.m.



(d) Champaign time 8 p.m.

Figure 2.6: Probability of bursty losses that cannot be recovered, $P(fail \mid loss)$ in left graphs and $P(fail)$ in right graphs, for different interleaving factors.

$P(fail \mid loss)$ can be derived from Equation (2.1), given an interleaving factor of i:

$$P(fail|loss) = 1 - \frac{\sum_{j=1}^{i-1}(jN_j) + i \times N_i \times \frac{i-1}{i}}{NL} \tag{2.2}$$

Further, $P(fail)$ can be computed as follows:

$$P(fail) = P(fail \mid loss) \times P(loss) \tag{2.3}$$

where

$$P(loss) = \frac{NL}{NP} \tag{2.4}$$

Figure 2.6 clearly shows that both $P(fail \mid loss)$ and $P(fail)$ drops quickly when the interleaving factor increases. For all times and all six connections, $P(fail)$ is negligible when the interleaving factor is equal to or greater than 4. Moreover, except for the China-Champaign connection, an interleaving factor of 2 works well for all the other five connections, achieving $P(fail)$ well below 3%. However, an interleaving factor of 2 is not always enough for the China-Champaign connection (see the graphs on the right in Figures 2.6(a) and 2.6(b)), because about 10-15% of the total losses cannot be recovered.

From the above experimental results, we conclude that a small interleaving factor (between 2 and 4) is adequate. In most cases, an interleaving factor of 2 leads to good recovery. Moreover, these interleaving factors do not increase end-to-end delays significantly. Assume an interleaving factor of i and that each packet covers a sampling period of T ms, the additional end-to-end delay introduced by interleaving is $(i-1) \times T$ at both ends without considering the effect of jitter buffers. With jitter buffers included at the receiver, the buffering delay at the receiver will increase little for $i \leq 4$ (see Section 3.4); hence, the additional end-to-end delay will be a little more than $(i-1)T$ for small i. For example, the additional end-to-end delay is around 32 ms if the interleaving factor is 2 and each packet covers a sampling period of 32 ms.

2.3.3 Relationship among loss behavior, sending rate, and packet size

As pointed out by Comer [18], congestion is a condition of severe delay caused by an overload of datagrams at one or more switching points. When congestion occurs, delay increases, and a router begins to enqueue datagrams until it can route them. Each router only has finite storage capacity and can buffer only a finite number of datagrams. After the router reaches its capacity, datagrams received are dropped. To avoid aggregating congestion, transmission rates must be reduced when congestion occurs. This argument naturally leads to a conclusion that a lower sending rate results in a lower packet-loss rate. In this group of experiments, we first test the relationship between loss rate and sending rate.

In our experiments, we change the sending rate by varying the sending interval. The sending rate ranges from around 1 packet per second to around 100 packets per second. Figure 2.7 plots the loss behavior for different sending rates and times of day. For domestic connections, the loss percentage drops when the sending rate is lowered. For international connections, the correlation between loss percentage and sending rate is not obvious. To have a clearer view of how the loss percentage changes, we project the 3D graphs in Figure 2.7 to 2D graphs shown in Figure 2.8. It is clear that for the Stanford-Champaign and MIT-Champaign connections, the higher the sending rate is, the higher the loss will be. This is especially true when the sending rate is 100 packets per second. For international connections, lowering the sending rate does not necessarily result in lower losses. The difference between domestic connections and international connections may occur because international traffic needs to be routed through more gateways with different aggregate effects.

Next, we test the influence of packet size and sending rate on packet losses. Due to the limitation of echo services, packet sizes tested are between 5 and 512 bytes per packet. The results are shown in Figures 2.9 and 2.10 for China and Stanford, respectively. The different lines shown in both figures are the loss percentages for various packet sizes at different times of the day. From the figures, we cannot conclude any obvious relationship between packet size and loss rate. Small packet sizes result in either lower loss or higher loss. These graphs also demonstrate that decreasing the rate by several packets per second may not significantly reduce the loss rate. Moreover, the loss rate resulting from sending 100 packets per second is not significantly different from that of sending around 4 packets per second.

However, we cannot draw the conclusion that rate reduction is not useful when congestion occurs. Because the traffics we introduced were relatively small and because the switches that caused packet losses might be handling a huge amount of traffic, reducing the sending rate of our stream alone did not affect the overall traffic significantly. In contrast, if most of the streams routed through a congested switch reduce their rates, then congestion will surely be alleviated. Thus, rate adaptation should be adopted even though no obvious immediate gain can be achieved by each individual application.

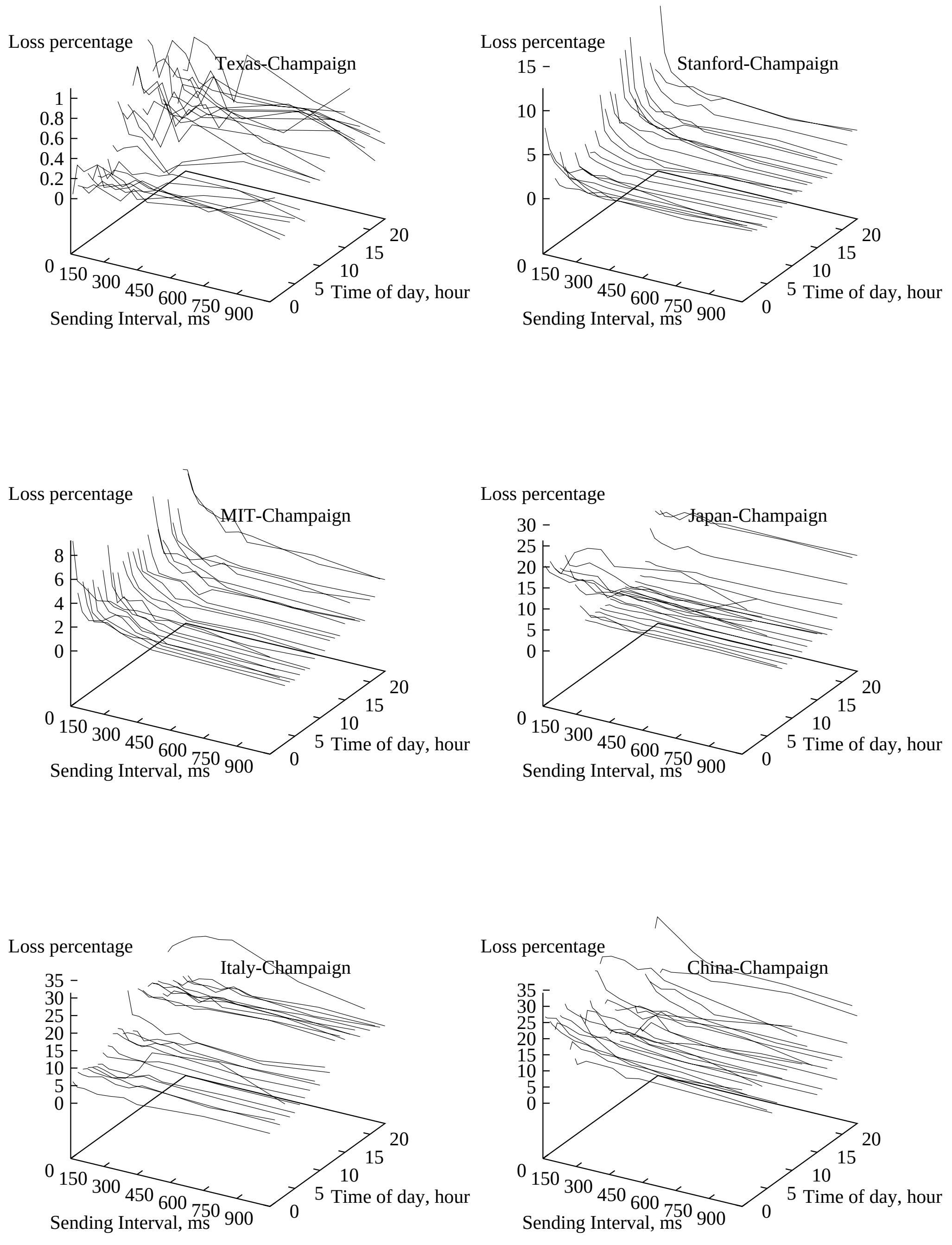


Figure 2.7: Loss percentage to different destinations under various sending intervals.

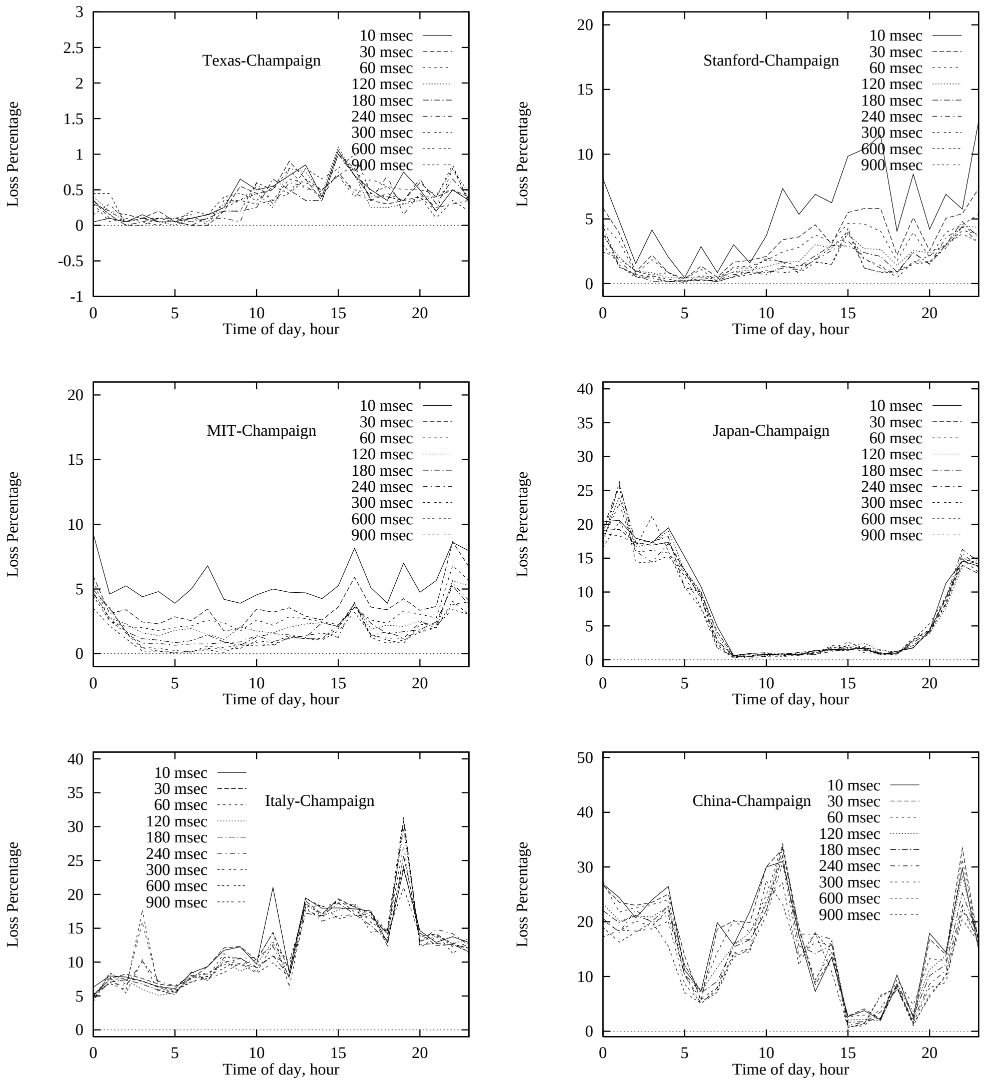
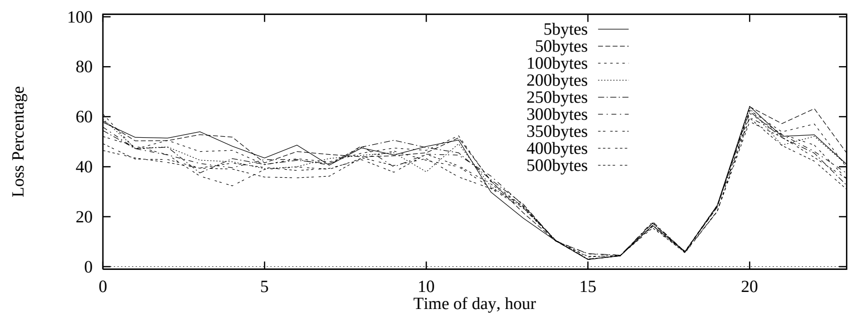
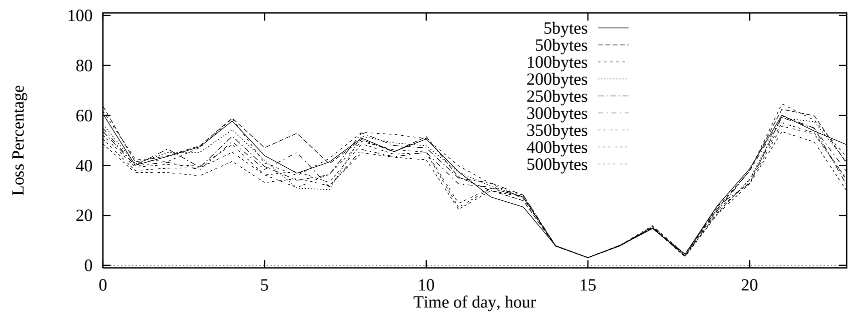


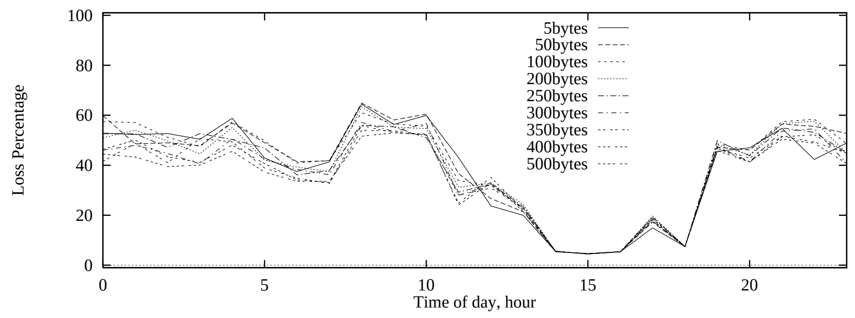
Figure 2.8: Loss percentage versus sending interval.



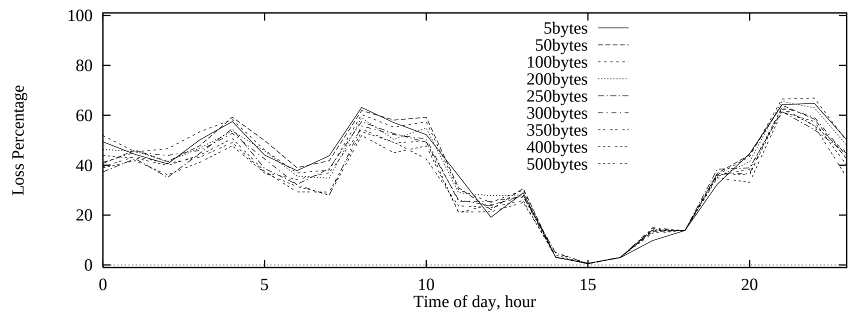
(a) Sending interval of 10 ms



(b) Sending interval of 120 ms

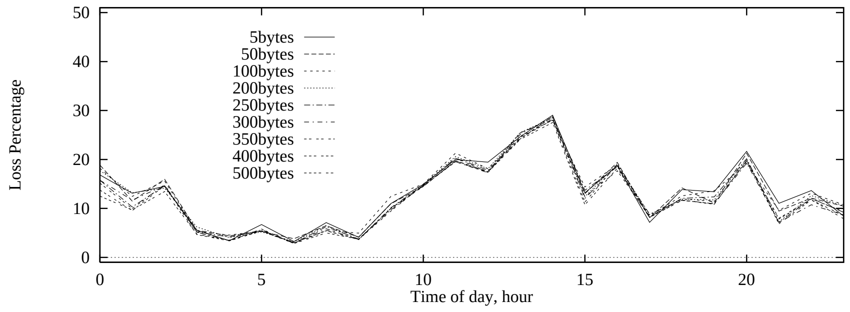


(c) Sending interval of 240 ms

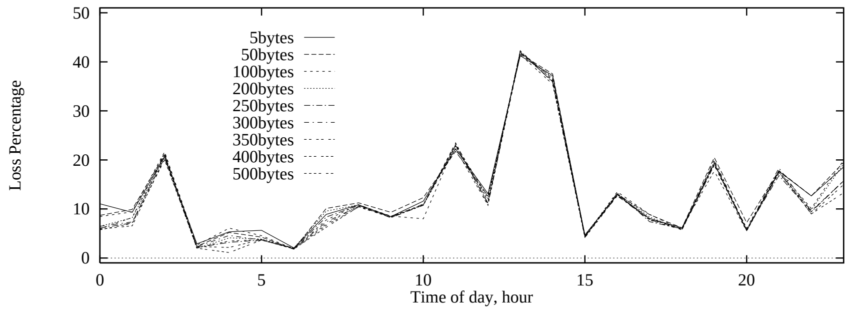


(d) Sending interval of 600 ms

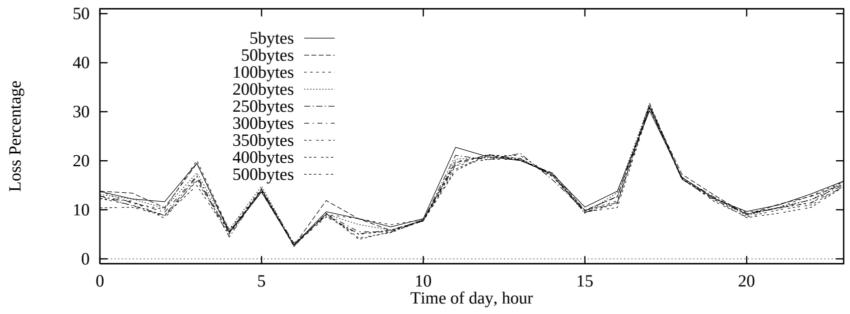
Figure 2.9: Loss percentage versus sending rate and packet size for the China-Champaign connection.



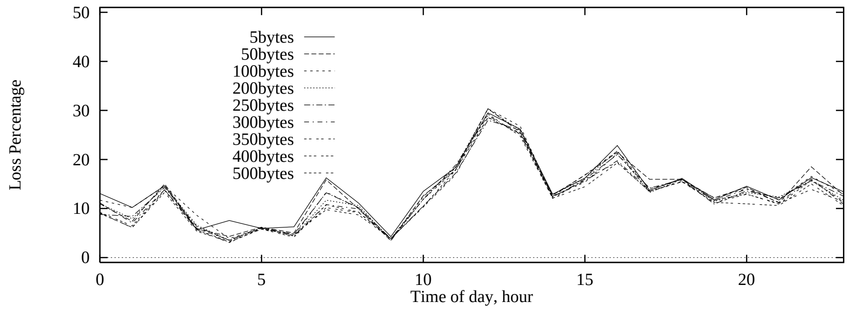
(a) Sending interval of 10 ms



(b) Sending interval of 120 ms



(c) Sending interval of 240 ms



(d) Sending interval of 600 ms

Figure 2.10: Loss percentage versus sending rate and packet size for the Stanford-Champaign connection.

2.3.4 Overall delay and jitter behavior

Delay and jitter are very critical to the quality of real-time voice transmissions. Figure 2.11 plots the round-trip-time PDF for both domestic and international connections. The results reveal that it is possible to satisfy the end-to-end delay (300 ms) in telephony-quality constraint if the connection is set up inside the US and if the processing times at both ends are kept low. The round-trip-time is only several tens of milliseconds for connections inside the US. The round-trip-time, however, increases dramatically for international connections, and will violate the delay constraint and degrade the quality of service for voice communications. Since an application has no way to control the transmission delay, it can only reduce the processing and buffering delays at both ends and hopefully achieve an acceptable voice quality.

Figure 2.11 also shows that the distribution of packet delays for a connection is highly concentrated around some value. Because jitter is several times larger than end-to-end delay, this means that even for sites with low delay, jitter can be really high. For example, for the MIT-Champaign connection, the average round-trip time is only tens of milliseconds; however, jitter may approach one second. Hence, in order for the voice delivery system to produce a continuous voice stream, jitter buffers are necessary. The trade-off between using larger jitter buffers and increasing end-to-end delay must be considered. The larger the jitter buffers are, the smoother the output stream and the larger the end-to-end delay will be. Hence, the system needs to decide on a suitable jitter-buffer size in order to achieve a certain continuity for the voice stream while maintaining a tolerable delay. In addition, Figure 2.11 shows that delay and jitter are also time-varying and connection-dependent, which imply that there is no single jitter-buffer size that is suitable for all conditions. A real-time system, therefore, should dynamically adjust its jitter-buffer size according to run-time statistics collected.

2.4 Summary

In this chapter, we have focused on studying the traffic patterns for voice transmissions over the Internet and on deciding the necessary parameters of a voice transmission system.

All the experimental results demonstrate that the traffic patterns are connection-dependent and time-varying. Because our system needs to adapt to a specific traffic pattern on the fly, a module that collects and analyzes statistics in real time is needed. This module will compute the loss rate, bursty loss length, end-to-end delay, and jitter of the current connection. This information will be

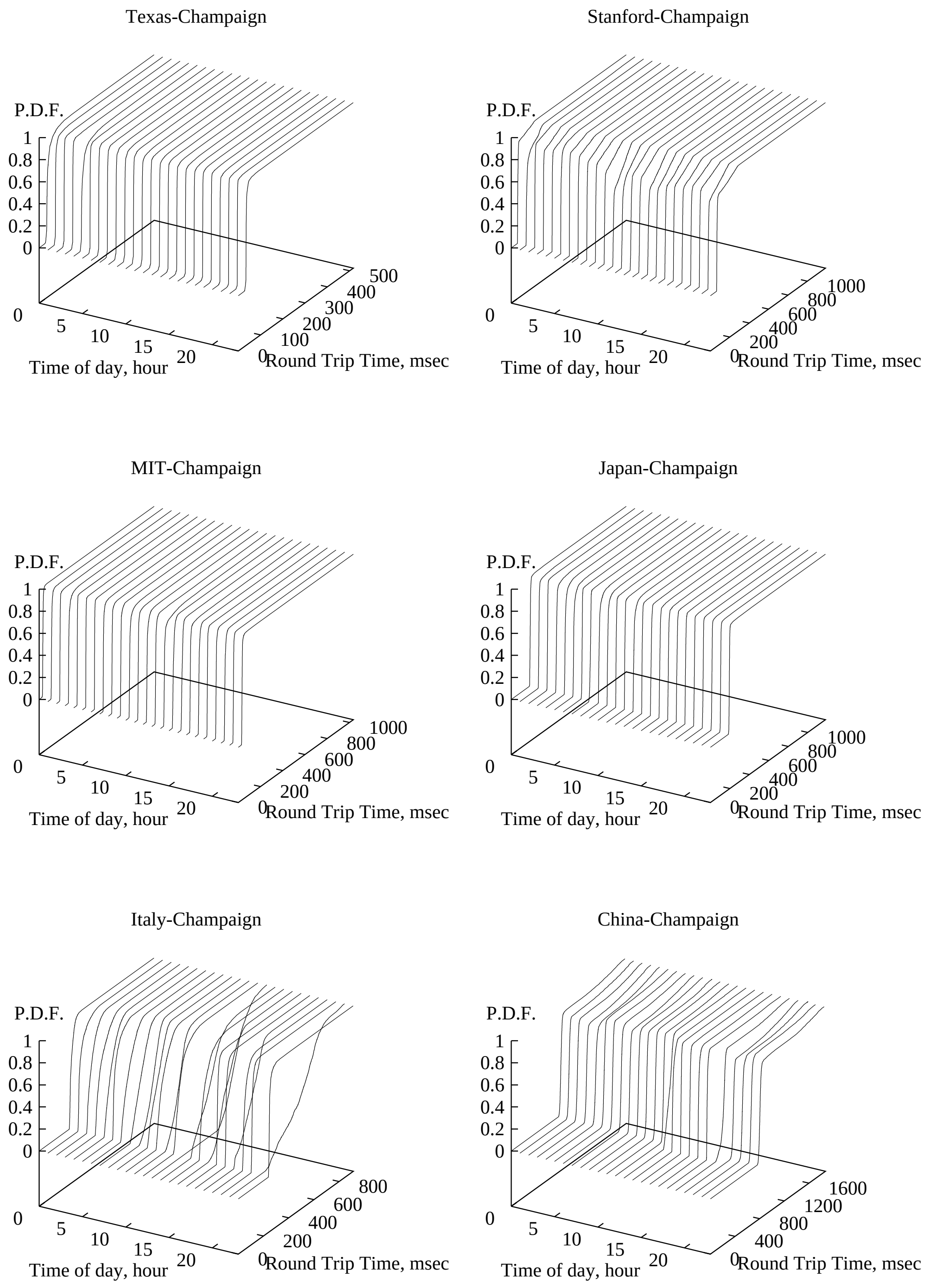


Figure 2.11: Round-trip time probability distributions.

fed back to the sender for the sender to decide how to overcome loss and satisfy delay constraint. Also, the information will be used to decide a suitable jitter-buffer size at the receiver.

Interleaving becomes a very attractive scheme to help overcome loss, as network loss happens randomly and the length of consecutive packet losses is relatively small. By revisiting Figure 2.3, we conclude that the interleaving factor does not have to be large: 2 is enough for domestic, and 4 is enough for international connections for most cases. Although interleaving will increase end-to-end delay by buffering more data at both ends, it has the advantage of utilizing the inherent redundancy in voice signals to overcome loss, without overloading the network. If both packet size and interleaving factor are kept small, end-to-end delay can be controlled in a reasonable range.

Rate adaptation may not be critical for each individual application; however, it is essential in making the whole network more predictable and controllable. The design of a rate-control module is difficult because immediate improvement may not always be seen by the application.

As stated in Section 2.3.4, jitter buffers are important in reducing "bubbles" in a received voice stream. The size of jitter buffers needs to be dynamically adjusted to suit the ever-changing behavior of a connection.

CHAPTER 3

SYSTEM DESIGN

This chapter presents our real-time voice transmission system that consists of silence detection, compression/decompression, interleaving, reconstruction, statistic collection, and buffering control (Figure 3.1). The basic ideas and functionalities of the different parts are explained. The modules dealing with loss, interleaving, and reconstruction, will be presented in more detail in Chapter 4.

3.1 Silence Detection

The first important component of the system is silence detection and suppression. It is an essential building block of any voice transmission system because it reduces the bandwidth needed in the underlying network transport service and helps maintain an acceptable end-to-end delay.

The purpose of silence removal is to identify and remove long runs of silence from a voice signal stream. It is well known that during an average conversation, each subscriber speaks only $40-50\%$ of the time [19]. A voice transmission system is not concerned with removing intersyllabic or interword silences. In fact, removal of these silences would degrade the quality of speech due to the temporal nonlinearity. It would create an effect of talking very quickly since the syllables and words would run very closely to each other. Studies have shown that 99.56% of continuous speech segments have periods of silence smaller than 150 ms [20]. For this reason, at an 8 kHz sampling rate, removal of a block of signals less than 1200 samples would reduce the quality of speech. Also, the performance gained by removing interword and intersyllabic silence would be quite small in comparison to the substantial removal of $50-60\%$ of real silence in signals, which will not affect the perceived quality of voice [21].

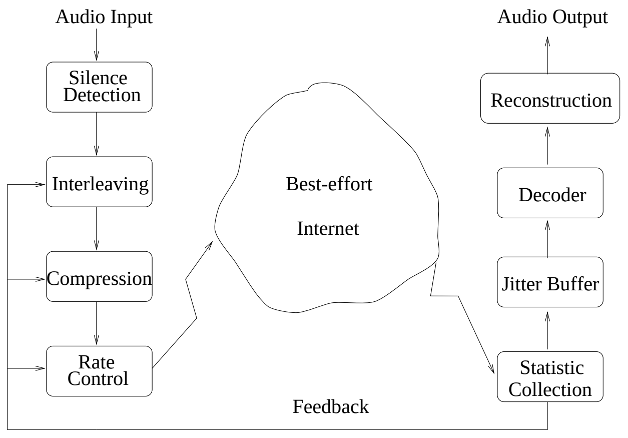


Figure 3.1: System architecture.

One popular class of algorithms is based on two measures of speech: short-time energy and zero crossing rate [22]. This class of algorithms is somewhat self-adapting to a background acoustic environment in the sense that it obtains all the relevant thresholds on its decision criteria from measurements made directly on the recorded interval. These algorithms have low processing complexity, reliable location of significant voice activities, and applicability to a variety of background silences. Hence, we chose a general algorithm of this class to be our silence detector.

The detailed algorithm [23] is as follows. As mentioned in the last paragraph, the algorithm is based on two simple measures: energy and zero crossing rate. The speech energy $E(n)$ is defined as the sum of the magnitudes of a frame of speech centered on the measurement interval. The use of magnitude de-emphasizes large-amplitude speech variations and produces a smoother energy function. The zero crossing rate of a piece of speech $z(n)$ is defined as the number of zero crossings per frame. Although the zero crossing rate is highly susceptible to 60 Hz hum, dc offset, etc., it is a reasonably good measure of the presence or absence of voice speech in most cases. A set of signals are first passed through a high-pass filter to get rid of low frequency hum. Next, the statistics of background silence are measured, assuming that there is no speech during the first 100 ms of the recording interval and that the threshold of silence energy and zero crossing rate are set [24]. Finally, the processing of voice signals can take place.

It is obvious that the removal of silence will decrease bandwidth requirements. Also, the removal of silence does not necessarily increase end-to-end delay since it decreases the processing time of other procedures at both the sender and receiver sites.

3.2 Compression

The aim of voice compression in the delivery system is to reduce bandwidth usage. The general function of all speech coding schemes is to analyze the signal, remove redundancies, and efficiently code the nonredundant parts of the signal in a perceptually acceptable manner. As the coding bandwidth is reduced, strategies for redundancy removal and bit allocation need to be even more sophisticated [25]. Since our work does not focus on the design of compression algorithms, we just choose one that is suitable for our system. To choose an appropriate compression algorithm, we have surveyed current standards about compressing voice data for multimedia systems. There are two broad categories: waveform coder and vocoder [26].

Waveform coders are characterized by their attempt to preserve the general shape of the signal waveform. As such, waveform coders are not speech specific in the sense that they work on any input waveform bounded by certain limits in amplitude and bandwidth. By preserving the general outline of the signal waveform, these coders generally operate on a sample-to-sample basis, and their performance is effectively measured in terms of signal-to-noise ratio (SNR), as quantization is the major source of distortion in the output waveform. The success of waveform coding for speech has been limited to rates above 16 kb/s, but waveform coders are nevertheless very popular and will remain so due to their simplicity and ease of implementation. The first worldwide speech encoding standard, International Consultative Committee on Telephony and Telegraphy's (CCITT's) 64 kb/s pulse-code modulation (PCM) with logarithmic quantization of is a waveform coder, and the coded speech is generally used as the reference for comparing lower-rate speech coders, as its performance is considered to be *toll quality.* Adaptive differential PCM (ADPCM), International Telecommunication Union (ITU) standard G.727, employs adaptive prediction and adaptive quantizers to exploit redundancies in speech signals and can compress speech signals to 40, 32, 24, or 16 kb/s.

At the opposite extreme to waveform coders, vocoders are very speech specific in their principles, because no attempts are made to preserve the original speech waveform. Waveform coders operate in the time domain, whereas vocoder mainly operates in the frequency domain. A generic vocoder consists of an analyzer and a synthesizer. The analyzer at the sender extracts from the original speech a set of parameters representing the speech production model, which characterizes the resonance frequency spectrum and the vocal tract excitation source. At the receiver, the speech is synthesized using the parameters to produce an often crude and synthetic reconstructed speech signal. One great advantage of vocoders is their low bit rate. For example, ITU standard G.723.1

(CELP) can operate at 5.3 or 6.3 kb/s and US Department of Defence's (DoD's) standard linear prediction coder (LPC-10) can operate at as low as 2.4 kb/s. As expected, SNR distortion measures are useless for vocoders; hence, subjective measures are generally used, such as mean opinion scores (MOS) test, diagnostic rhyme test (DRT), and diagnostic acceptability measure (DAM) [27]. As experiments indicated [27], vocoders can produce highly intelligible speech, but are let down by their inability to faithfully reproduce a speaker's characteristics and their susceptibility to background noise.

To select a suitable speech coder, we consider four aspects of speech compression algorithms: *bit rate, complexity, delay, and quality* [28].

Bit rate is the communication-channel bandwidth at which the coder operates. If, by obeying the umbrella standards H.323 for multimedia transmissions, G.723.1 is chosen as our speech coder, which already incorporates silence compression [29], its bandwidth usage can be as low as 5.3 kb/s. In contrast, if G.727 is chosen, the bandwidth usage needs to be 16 kb/s, but can be reduced to about 8 kb/s after incorporating silence suppression. From this point of view, G.723.1 is better than G.727.

Complexity refers to the computational complexity of the speech coder. This has important consequences for most applications. Vocoders are generally more complex than waveform coders. For example, a G.723.1 implementation is estimated to require 14-20 fixed-point million instructions per second (MIPS), LPC-10, about 7 MIPS in a general-purpose digital signal processor (DSP), and G.727, much lower complexity [26] at about 2 MIPS [30]. One direct consequence of complexity is the computational delay the coder may cause.

Delay refers to the communications delay caused by the coder. Individual sample coders (waveform coders) have the lowest delay, whereas coders that work on a block or frame of samples (vocoder) have higher delays. Besides having serious repercussions on a conversation, too much delay creates critical challenges on a network echo canceler and forces speakers into an inconvenient "push-to-talk" mode, making conversation ineffective. Since the practical limit of one-way delay for telephony is about 300 ms, and after taking into account of transmission and other delays in the system, the delay constraint of speech coders is further restricted.

The software codec of G.723.1 we tested cannot encode voice signals in real time, as it needs approximately 60 ms to encode 32 ms of samples. If only sending small frames, the overhead of IP and UDP headers must also be taken into account. For instance, if G.723.1 frames are sent frame by frame, the payload of voice data is only 22 bytes (for 5.3 kb/s), with an overhead of at least 28 bytes. Hence, 64 ms of data will actually need 100 bytes, and the bandwidth taken is $(100 * 8/64)$

Sequence Num.	Time Stamp	Interleaving factor	Sending Rate

Figure 3.2: Additional packet header.

kb/s for an encoding delay of 120 ms. In contrast, a G.727 codec only needs 15 ms to encode 32 ms of samples. Further, the 64 ms data can be packed in one packet, resulting in $[(64+28)*8/64]$ kb/s for an encoding delay of 30 ms after silence removal. Hence, both schemes requires comparable bandwidth usage. Although the overhead of the underlying protocols will have less effects for larger packet sizes, the end-to-end delay restriction will be hard to satisfy for a G.723.1 coder. From the *complexity* and *delay* points of view, a G.727 coder is more suitable for us.

Quality refers to a large number of attributes. As bit rates are lowered, speech coders become more speech specific and give less faithful renditions of other sounds. Background noises, such as noise in an office, can all affect the perceived quality of a speech coder. Using MOS for quality comparison, ADPCM is about 4.1, CELP is about 3.2, and LPC-10 is about 2.3 [30].

In summary, we chose ADPCM as our current compression scheme, considering all the above factors for our software-oriented voice transmission system.

3.3 Statistics Collection and Rate Control

The characteristics of Internet traffic are changing constantly, as pointed out in Chapter 2. To make the different parts of a system work in a dynamic network, a statistic collector is needed, together with additional header space in each packet to facilitate the propagation of statistics.

The arrival of UDP packets at the receiver will be out-of-order, so sequence number needs to be put in each packet's header. To know the end-to-end delay, each packet is time stamped. The interleaving factor and the sending rate are also needed for the reconstruction module. Figure 3.2 shows the packet header sent by the sender.

At the receiver, the arrival time of each packet is also recorded. From the time stamp in each packet header and the arrival time, end-to-end delay and jitter can be estimated and fed into the jitter-buffer module. Note that packets are reordered according to their sequence numbers. Also, loss l can be estimated by the sequence numbers collected during a certain interval T as follows:

$$l = \frac{C(\vec{s})}{\max \vec{s} - \min \vec{s}} \tag{3.1}$$

where $\vec{s}$ is the set of sequence numbers received during the interval, and $C(\vec{s})$ is the total number of packets received during the interval. Burst lengths can be computed from the sequence numbers as well. The receiving rate r_r is estimated as follows:

$$r_r = \frac{C(\vec{s})}{T} \tag{3.2}$$

The loss, delay, and receiving rate are periodically fed back from the receiver to the sender by a UDP channel. The reason for using a UDP channel instead of a TCP channel is that we do not want the system to retransmit the feedback packets. If one feedback packet is lost, a new feedback packet carrying new information will be sent to the sender after one period of delay.

Congestion information, indicated by loss, delay, and bursty loss length, is used to adjust the interleaving factor at the sender. From our experiments, the interleaving factor need not be very large. For domestic connections, an interleaving factor of 2 works quite well, whereas for international connections, 2 or 4 is enough depending on locations, since the majority of burst length is within this range from our experiments shown in last chapter.

As stated in Chapter 2, rate control has no significant consequence for one stream of data; however, it still needs to be included. The rate-control method adopted here is a simple one. The basic idea is to periodically send back the loss rate, burst length, and delay statistics to the sender using a UDP channel. The sender then uses the information to adjust its sending rate. If losses are high, the sender reduces the sending rate, leading to down-sampled voice signals. If loss is low, the sending rate is increased to improve the transmission quality.

3.4 Jitter-Buffer Design

The purpose of a buffering algorithm at the receiver is to delay the first voice packet in a talk spurt by a *hold time* before it is played. If the playback time of the first packet is set, all playback times of the subsequent packets are set as well. This hold time is crucial since it determines the playback times for all subsequent packets and the playback quality. If it is set too large, there will be unnecessary end-to-end delay. If it is set too small, then the loss rate will increase due to some delayed packets, and the playback quality will deteriorate due to the burstiness of the received voice stream.

The criteria of the hold time is to ensure positive slack for each packet, namely,

$$p_i - a_i > 0 \tag{3.3}$$

where p_i and a_i are the playback and arrival times of packet i. The hold time $p_1 - a_1$ is calculated using the following equation [4]:

$$p_1 - a_1 = t_1 + \hat{d} + 4 * \hat{v} - a_1 \tag{3.4}$$

where t_i is the transmission time of the ith packet, and $\hat{d}$ and $\hat{v}$ are estimates of the mean and variance of end-to-end delay. Several methods for network delay estimation have been proposed [4, 5, 10, 31, 32, 33]. *Algorithm 4*, proposed by Ramjee et al. [4], develops an estimator that explicitly considers the effects of sudden large changes in delay, called *delay spikes*, and can achieve a lower rate of lost packets for both given average playback delay and maximum buffer size.

If the effect of interleaving is considered, then the above criteria in Equation (3.3) must be changed. Assuming an interleaving factor of f, the first packet in an interleaving set needs to wait for the following $f - 1$ packets to arrive in order to be de-interleaved. Hence, the criteria should be changed to

$$p_i - a_{i+f-1} > 0, \; for\ some\ packet\ i \tag{3.5}$$

The corresponding hold time $p_1 - a_1$ will be increased as well:

$$p_1 - a_1 = t_1 + \hat{d} + 4 * \hat{v} - a_1 + \lambda \times (f - 1) \times (t_i - t_{i-1}) \tag{3.6}$$

where $t_i - t_{i-1}$ is the sampling period of packet i. If all packets need to wait for $f - 1$ packets before playing, then the hold time in Equation (3.4) should be increased by $(f - 1) \times (t_i - t_{i-1})$. However, since a fraction $(1/f)$ of packets need to wait that long, λ usually takes a value around 0.25.

3.5 Summary

In this chapter, we discussed the various components of our transmission system. In each component, we presented its motivations and its main functionalities.

To reduce bandwidth usage, both silence removal and compression are important. Because our system is software based, the compression scheme chosen needs to be of low delay. These considerations lead us to adopt the G.727 codec. The bandwidth after silence removal and compression using G.727 is about 8 Kbps.

A statistics collector is used to calculate loss, delay, jitter, etc., and feeds the information to other components in order to help adjust system parameters. A rate controller is used to choose an appropriate sending rate, depending on current available bandwidth.

Interleaving and reconstruction components are used to handle loss that will be explained in the following chapter.

Jitter buffers are important for achieving high playback quality and for balancing end-to-end delay and discontinuity in the received voice stream. By using information on delay and jitter from the statistics collector, our system estimates the holding time of the first packet for a talk spurt.

Currently, we have implemented the following in our prototype: silence detection, compression/decompression, jitter-buffer control (with fixed parameters), interleaving (with fixed factor depending on destination), and reconstruction. We plan to implement in our future prototype rate-based control and dynamically adjusted parameters for different parts of the system according to network conditions.

CHAPTER 4

RECONSTRUCTION

We introduce two new methods in this chapter based on interleaving and average reconstruction. For simplicity, consider two-way interleaving (see Figure 1.1), in which the sender first divides the voice stream into two substreams, one with even samples and the other with odd samples. The sender then groups these substreams into packets and sends them over the Internet. The two packets with the corresponding even and odd samples are called an *interleaving pair*. In case of no loss, the receiver receives both substreams and de-interleaves the packets into the original signals. In case that one packet in an interleaving pair was lost, the receiver reconstructs each lost sample by taking the average of its two adjacent samples received.

Assume the input voice stream to be $x_0, x_1, \cdots$, the size of each packet to be N samples, packet P_1 with even samples $x_0, x_2, \cdots, x_{2N-2}$, and P_2 with odd samples $x_1, x_3, \cdots, x_{2N-1}$. If the receiver receives P_1 but not P_2, then P_2 is reconstructed as follows:

$$\hat{x}_{2j-1} = \frac{x_{2j-2} + x_{2j}}{2} \qquad \text{where } j = 1, \ldots, N \tag{4.1}$$

The interleaving and average-reconstruction method described above is a best-effort method with no guarantee on quality. To improve over this method, we first design an adaptive filter-based reconstruction algorithm on top of average-reconstruction method to further decrease reconstruction errors. Second, we employ a new transformation-based reconstruction algorithm to reduce large reconstruction errors caused by rapidly changing parts in voice signals. In the following sections, we discuss these two new reconstruction methods.

The experiments reported in this chapter were carried out by using four sound files listed in Table 4.1. All files are of 16-bit linear PCM format with sampling rate of 8000 samples per second for convenience of processing.

Table 4.1: Audio files used in the experiments.

File	Originality	Length (kb)	Characteristic
1	www.geocities.com/Hollywood/Hills/6498/ateyours.wav	127.4	movie clip (talk)
2	www.geocities.com/Hollywood/Hills/6376/heat.wav	115.6	movie clip (man)
3	admii.arl.mil/~fsbrn/phamdo/speech_demo.html	65.6	woman's speech
4	recorded from microphone	1797.9	music

4.1 Adaptive-Filter-Based Reconstruction Method

In this section, we first introduce adaptive filtering implemented on top of average reconstruction in order to achieve a better reconstruction quality. We then discuss the application of the adaptive filtering algorithm in the reconstruction process.

4.1.1 Introduction to adaptive filtering and LMS approach

An adaptive system is a device with continuous-range parameters that are adjusted under the influence of a "reference signal" in order to produce this particular reference output in answer to a specific input [34]. The adaptive process is actually an optimization process with the following properties: (a) no prior knowledge of the optimal parameters are assumed, (b) the adaptive filter can constantly and automatically adjust itself to approach the optimal parameters.

Conventional, nonadaptive filters used to extract information from input signals are normally linear time invariant. That is, they perform exactly the same set of linear operations on all signals irrespective of time [35]. In case of adaptive filters, this restriction on time invariance is removed. This is done by allowing the filter to change its coefficients used in the linear filtering operation according to some predetermined optimization criteria. Hence, if we do not know the optimal filter coefficients beforehand, an adaptive filter can be used to find the best coefficients step by step on its own.

In a generic adaptive filtering problem, two time sequences are observed: x_k and d_k (see Figure 4.1), where x_k is the input of the adaptive filter. The corresponding output sequence, y_k, is calculated as follows:

$$y_k = \sum_{l=0}^{l=M-1} w_l * x_{k-l} \tag{4.2}$$

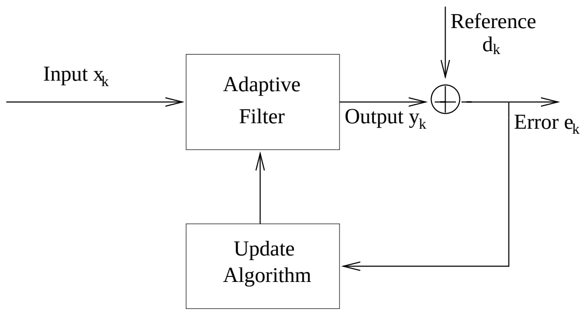


Figure 4.1: An adaptive filter.

where $\vec{w} = w_0, \cdots, w_{M-1}$ represent coefficients of a filter of length M. Because the filter coefficients w_l equals zero for l less than zero, the filter has causal impulse response. Hence, the analysis in this section is causal analysis. It is desirable to choose $\vec{w}$ such that output y_k is as close as possible to d_k, the reference. The output error

$$e_k = d_k - y_k \tag{4.3}$$

should be made as small as possible. This is usually done by minimizing the mean square error (MSE) $E[{e_k}^2]$. Because the process of statistical averaging in adaptive filtering smoothes out variations in sequences x_k and d_k, $E[{e_k}^2]$ is only dependent on filter parameters $\vec{w}$. The problem is, therefore, to find $\vec{w}_{opt}$ that minimizes $E[{e_k}^2]$; that is,

$$\vec{w}_{opt} = \min_{\vec{w}} E[{e_k}^2] \tag{4.4}$$

This vector $\vec{w}_{opt}$ represents the least mean square (LMS) filter that may be time varying.

Mathematically, the optimal coefficients can be computed from Equation (4.4) by substituting e_k using Equation (4.3) and further substituting y_k using Equation (4.2). The result is

$$\vec{w}_{opt} \quad = \quad R^{-1}\vec{p} \tag{4.5}$$

where

$$R = E[\vec{x}\vec{x}^t] \tag{4.6a}$$

$$\vec{p}^t = [r_{dx}(0), ..., r_{dx}(M-1)] = [E[d_k x_k], ..., E[d_k x_{k-(M-1)}]] \tag{4.6b}$$

The solution is unique when R is positive definite. The problem with this approach is that, although it is great in theory, in practice the statistics of x_k and d_k is not known. One way to surmount this problem is to estimate the statistics adaptively and track changes of signals over time to keep the system near optimal all the time. However, this method requires computing a matrix inverse after receiving each sample. Although observing that R is Toeplitz can save some computation time, the approach is still too expensive for real-time applications.

Besides this direct approach, there is another approach that solves the problem by utilizing the fact that the error surface $E[{e_k}^2]$ is a unimodal "bowl" in space R^M if R is positive definite. (The shape and bottom of the bowl may drift slowly with time, because we are dealing with nonstationary speech signals.) To find the bottom of the bowl, gradient descent methods can be used. The idea is to iteratively find the minimum by computing gradient ∇ of the error function.

$$\nabla = \frac{\partial E[{e_k}^2]}{\partial \vec{w}} \tag{4.7}$$

Gradient ∇ is a vector in space R^M pointing to the steepest uphill direction on the error surface at a given point. Updating the coefficient vector by taking a step opposite to the gradient direction at time i amounts to going "downhill" in the steepest direction, which is a sensible way to iteratively find the minimum:

$$\vec{w}^{i+1} = \vec{w}^{i} - \mu \nabla^{i} \tag{4.8}$$

The performance of the algorithm in Equation (4.8) obviously depends on μ. If μ is too large, the search will bounce back and forth, making the algorithm hard to find the minimum. If μ is too small, it may take a long time to approach the bottom, or may never approach the bottom if the bowl is time-varying.

Parameter μ is usually chosen based on time-invariant analysis [36]. There are two convergence criteria. First, *convergence in the mean*, defined as follows:

$$\lim_{i \to \infty} E[\vec{w}^{i} - \vec{w}_{opt}] = 0 \tag{4.9}$$

requires

$$\mu \quad < \quad \frac{1}{M \times R(0)} \tag{4.10}$$

where M is the filter length and $R(0) = E[x_k x_k]$. Only knowing that the estimated value will approach the optimal value in the average sense cannot guarantee that MSE is finite. Hence, *convergence in the mean square*, defined as follows:

$$\lim_{i\to\infty} E[|\vec{w}^i - \vec{w}_{opt}|^2] = 0 \tag{4.11}$$

needs to be satisfied too if a predictable behavior of the filter is desired. It requires

$$\mu \quad < \quad \frac{1}{N \times R(0)} \tag{4.12}$$

where N is the number of samples used to estimate $E[{e_k}^2]$.

A simplified and popular version of the gradient descent algorithm uses ${e_k}^2$ as an estimate of $E[{e_k}^2]$ [37]. This simplified LMS adaptive filter algorithm can be summarized as

$$y_k = \sum_{l=0}^{l=M-1} w_l * x_{k-l} \tag{4.13a}$$

$$e_k = d_k - y_k \tag{4.13b}$$

$$\hat{\nabla}^k = \frac{\partial {e_k}^2}{\partial \vec{w}} = -2e_k\vec{x}^k \tag{4.13c}$$

$$\vec{w}^{k+1} = \vec{w}^k - \mu\hat{\nabla}^k = \vec{w}^k + 2\mu e_k\vec{x}^k \tag{4.13d}$$

The LMS algorithm is often called a "stochastic gradient" algorithm, since $\hat{\nabla}^k$ is a noisy gradient. This is by far the most commonly used adaptive filtering algorithm, because it is simple, works well in practice, and requires relatively little computations [36]. In the next subsection, we apply this algorithm to the reconstruction phase of our system.

In summary, an adaptive filtering algorithm is able to "intelligently" estimate the reconstructed signals based on history information of the signals. By using this property, our reconstruction subsystem will be able to extract signal-specific information and use it to estimate missing signal values.

4.1.2 Adaptive-filter-based reconstruction for two-way interleaving

Before we combine adaptive filtering with average reconstruction, we consider voice signals in more details. A typical segment (8 samples) of voice signals in Figure 4.2 shows that digital speech signals are not smooth but are very rugged. In this figure, if all the odd samples are lost, then average reconstruction gives very poor recovery of the original waveform. Unlike the average

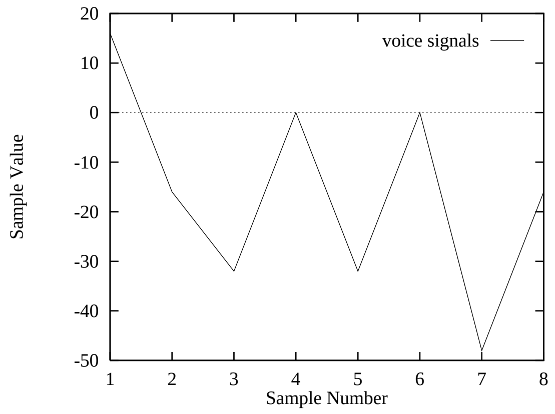


Figure 4.2: A segment of voice signals.

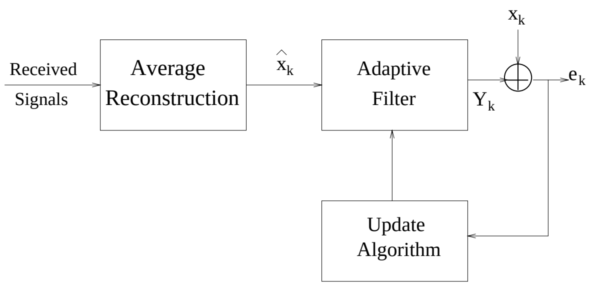


Figure 4.3: Adaptive-filter-based reconstruction.

reconstruction that always performs the same interpolation no matter what the reconstruction error is, an adaptive filtering algorithm can automatically adjust its interpolation to reduce the reconstruction error of future samples, based on the results of past samples.

In this subsection, we investigate the application of adaptive filtering on top of average reconstruction (Figure 4.3) in order to achieve a better reconstruction quality. The average reconstructed stream $\vec{\hat{x}}$ is fed into an adaptive filter to generate output $\vec{y}$, with a goal that $\vec{y}$ should be as close to $\vec{x}$ as possible. Given $\vec{e}$, the difference between $\vec{x}$ and $\vec{y}$, we like to minimize the reconstruction error in the average sense:

$$E[{e_k}^2] \quad = \quad E[(x_k - y_k)^2] \tag{4.14}$$

We cannot directly use the above formulation in applying the LMS adaptive filtering algorithm, because the original signals x_k are not always available at the receiver. When x_k is lost, the reference signal must be defined. Because the average reconstructed signal is an unbiased approximation to the desired signal, it is chosen as the reference. Also, one possible group of initial coefficients is $\vec{w} = 0, \cdots, 0, 0.5, 0, 0.5, 0, \cdots, 0$, based on average reconstruction. However, the analysis in the last subsection cannot be used directly because it is based on causal analysis, where $w_l = 0$ when $l < 0$. The formulation needs to be changed to noncausal, where $w_l \neq 0$ when $-(M-1)/2 \leq l \leq (M-1)/2$, if average interpolation is used as our initial coefficients. Output $\vec{y}$ should be computed as follows (where M is odd):

$$y_k = \sum_{l=-(M-1)/2}^{l=(M-1)/2} w_{l+(M-1)/2} * x_{k-l} \tag{4.15}$$

and the optimal coefficients can be obtained as:

$$\vec{w}_{opt} = R^{-1}\vec{p} \tag{4.16}$$

where

$$R = E[\vec{x}\vec{x}^t] \tag{4.17a}$$

$$\vec{p}^t = [r_{dx}(-(M-1)/2), ..., r_{dx}((M-1)/2)] \tag{4.17b}$$

Compared to Equations (4.5) and (4.6), $\vec{w}_{opt}$ and R have the same form whereas $\vec{p}$ is changed. The convergence analysis of μ will not change because it only depends on covariance matrix R that does not change. The adaptive filter of this form is symmetric to the value currently being estimated. That is, it not only uses past samples to estimate the current value but also uses future samples. Although future samples may not be available for some applications, they are available in our case, since the signals are transmitted packet by packet.

The process of applying the LMS adaptive filtering algorithm after average reconstruction is to determine the appropriate parameters of the adaptive filter in order to achieve better reconstruction quality. The effects of different parameters are studied by experimenting with real audio files mentioned at the beginning of this chapter.

The major factors affecting the convergence of the adaptive filter and its long term reconstruction quality are the filter length and its adaptation step. Filter length M determines the number of adjacent samples needed in the reconstruction of the lost sample and indirectly affects the efficiency of the algorithm. For efficient operations, we want M to be as small as possible, although the reconstruction quality may suffer when M is small. Figure 4.4 shows the relationship between filter length and improvements in reconstruction quality for various adaptation steps μ. The incomplete lines in Figure 4.4(a) for $\mu = 10^{-9}$ and Figure 4.4(c) for $\mu = 10^{-10}$ and the missing lines in Figures 4.4(b)-4.4(d) for $\mu = 10^{-9}$ are due to the divergence of the adaptation algorithm. Figure 4.4 shows that there is no single best filter length that works for all voice signals. The best filter length should be at the point that its corresponding μ is just on the border between convergence and divergence, leading to the fastest adaptation. Hence, the best filter filter length depends on the statistical characteristics of the input signals, since its corresponding μ depends on R.

Figure 4.4 has already shown the difference among various adaptation steps μ, where μ controls the convergence of the algorithm. Since our experiments were done on 16-bit linear PCM wave-formatted speech signals, the signals are in the range $[-32768, 32767]$. From the analysis in the last subsection, μ needs to be around 10^{-10} if we estimate $R(0)$ using 32767^2. The effect of different μ can be seen more clearly in Figure 4.5. When μ is larger than 10^{-10}, most of the SNR decreases dramatically because μ is too large for the filter to converge. When μ is smaller than 10^{-11}, SNRs decrease slowly because μ is too small to track changes in the signals quickly. Also, we see that different voice files have different convergence property, which is consistent with our analysis on filter length.

For real-time transmissions, it may be difficult to determine in real time the most suitable parameters for the current voice stream. Hence, the system should use a set of universal parameters. From Figures 4.4 and 4.5, we conclude that μ of 10^{-11} and filter length of 10 or more are feasible for all the four streams. Table 4.2 compares the reconstruction quality between with and without adaptive filtering using filter length M of 13 and μ of 10^{-11}. For these audio files, around $0.1 - 1$ dB improvements can be achieved over the case without adaptive filtering. We also observe that for file 3, the improvement is minimal. The main reason is that adaptive filtering needs time to adapt to changes in the signals. If the original signals are changing too fast, then adaptive filtering may have difficulty to keep track of trends. In order to deal with this situation, the sender needs to smooth the signals sent in order to improve the reconstruction quality. This idea leads us to design the following transformation-based reconstruction algorithm.

4.2 Transformation-Based Reconstruction

In this section, we propose a new sender- and receiver-based reconstruction algorithm that transforms input data so that distortions are minimized when some interleaved packets are lost and the missing samples are reconstructed using a given interpolation-based reconstruction method.

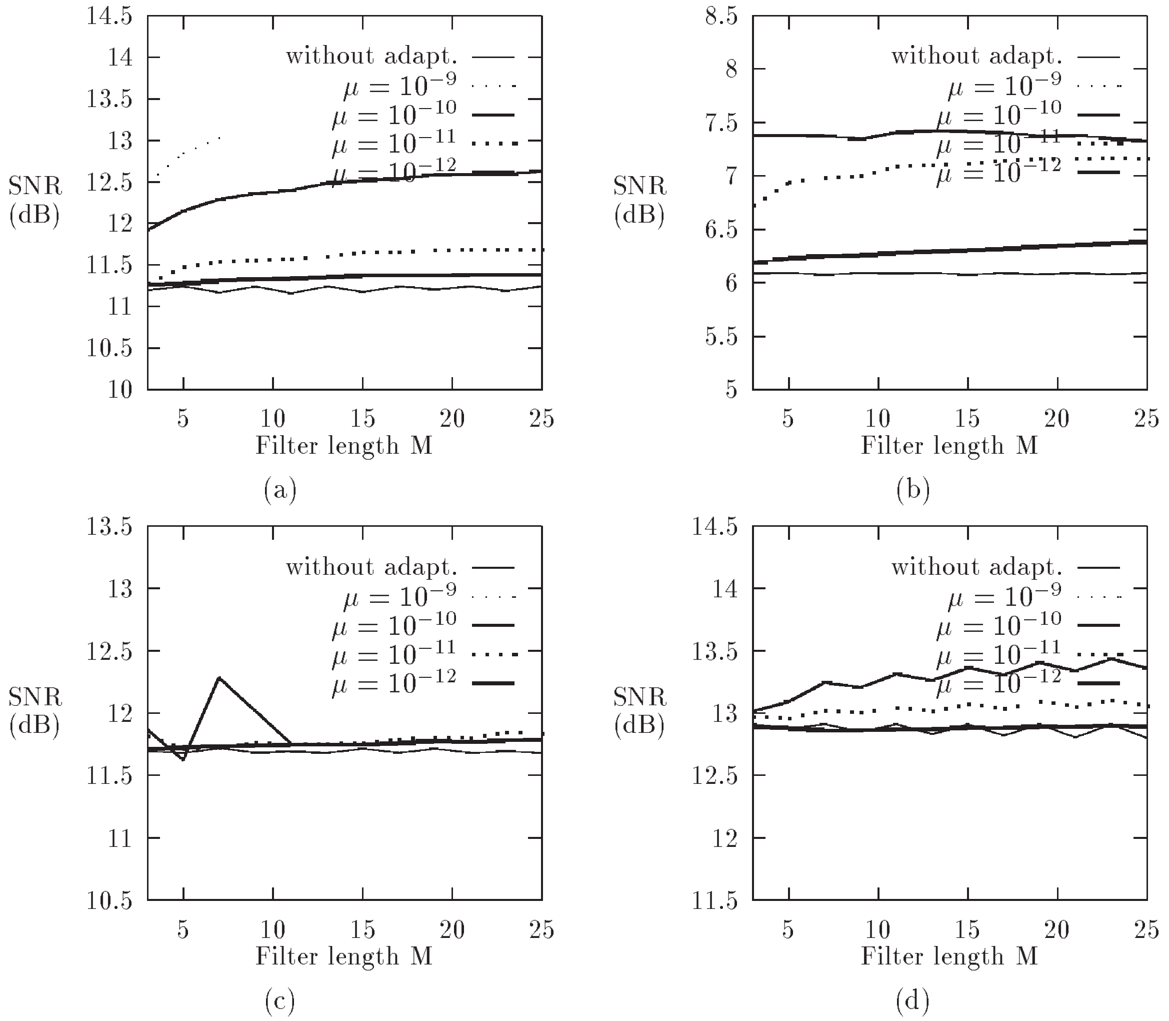


Figure 4.4: Comparison of reconstruction quality between that with adaptive filtering of different filter lengths M and that without on (a) a movie clip of a talk, (b) a movie clip of a man's voice, (c) a segment of woman's speech, and (d) a segment of music.

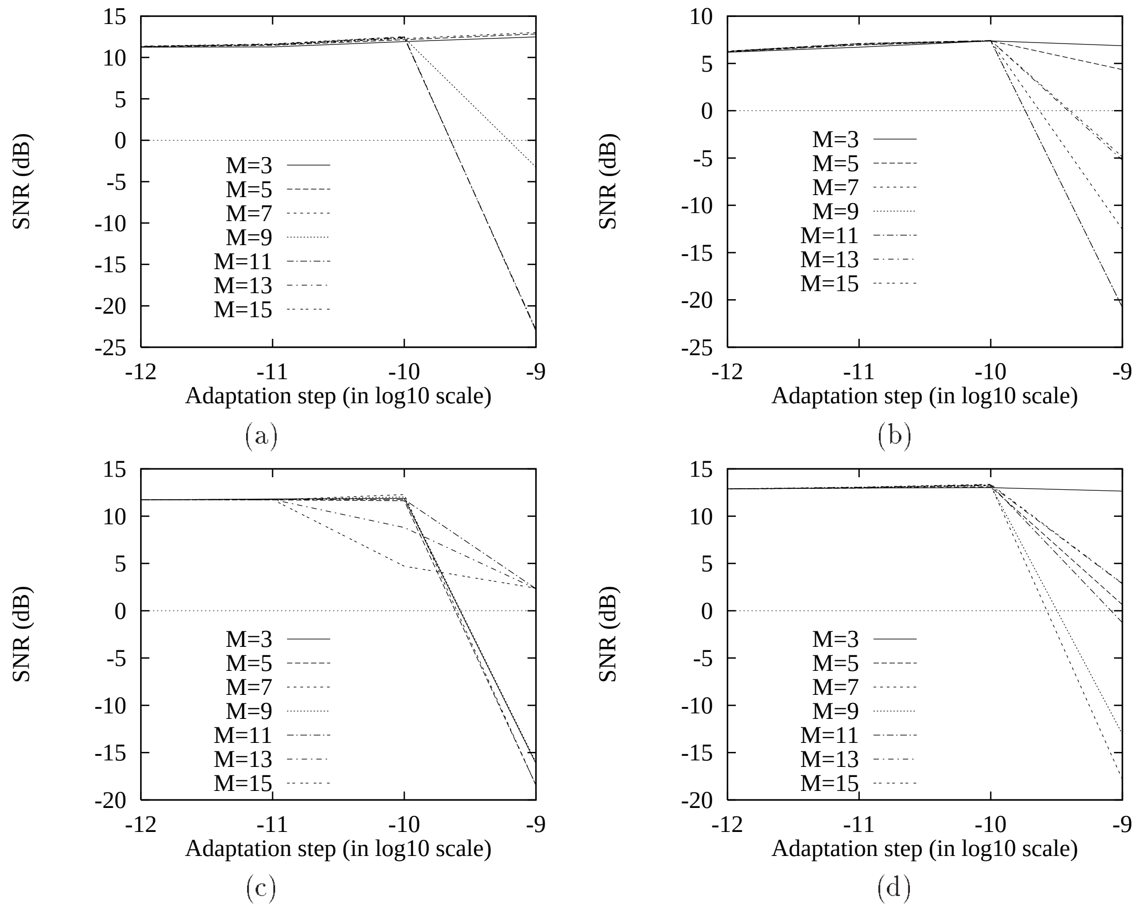


Figure 4.5: Effects of adaptation step size μ on (a) a movie clip of a talk, (b) a movie clip of a man's voice, (c) a segment of woman's speech, and (d) a segment of music.

Table 4.2: Reconstruction quality improvements for four different sound files with and without adaptive filtering. The first three files contains voice signals and the fourth one contains music signals. The adaptive filter uses a filter length of 13 and a adaptation step size of 10^{-11}.

Sound	SNR (dB)	
File	With	Without
1	11.60	11.24
2	7.10	6.09
3	11.75	11.68
4	13.02	12.83

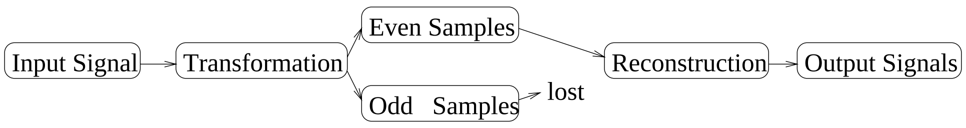


Figure 4.6: The process of transformation and reconstruction in two-way interleaving.

4.2.1 Transformation-based reconstruction for two-way interleaving

In this subsection, we formalize the reconstruction problem and derive the optimal transformation needed by the sender. For convenience, we analyze the problem for 2-way interleaving and a reconstruction algorithm based on the average of adjacent samples (Figure 4.6).

In the average reconstruction method, the sender performs interleaving and packetizes related information into different packets, hence preventing an isolated loss to cause the loss of an entire segment of information. Intuitively and experimentally, the method improves the transmission quality in the presence of isolated losses. However, it does not guarantee the quality of the reconstructed signals because the waveform of voice signals can sometimes be very rugged, and an average approximation may be inaccurate.

To overcome this problem without increasing network bandwidth, the sender transforms each original sample into a new sample before decomposition, interleaving, packetization, and transmission. It performs the transformation in such a way that the reconstructed samples at the receiver will be the best approximation to the original ones on the average, where the reconstruction criterion is defined as the MSE between the reconstructed signals and the original signals.

Although our reconstruction algorithm is based on the average of adjacent samples, our optimization process can be easily adapted to any other interpolation-based reconstruction methods.

There are two cases to be considered at the receiver, the first case is when only one of the packets in an interleaving pair is received, and the other when both packets in the interleaving pair are received.

Case I: One packet loss in an interleaving pair

Suppose the original data stream at the sender, $\vec{x} = x_0, x_1, ..., x_{2N-1}$, is transformed into stream $\vec{y} = y_0, y_1, \cdots, y_{2N-1}$ by transformation $\mathbf{T}$ (unknown yet). Suppose further the receiver only receives half of the stream. Without loss of generality, assume all even samples $\vec{y}_{even} = y_0, y_2, \cdots, y_{2N-2}$ are received. After reconstruction using the average reconstruction method, the reconstructed stream, denoted by $\vec{\hat{y}}_{even} = \hat{y}_0, \hat{y}_1, ..., \hat{y}_{2N-1}$, is calculated as follows (assuming $x_{2N} = 0$):

$$\hat{y}_i = \begin{cases} y_i, & i \text{ even} \\ \frac{y_{i-1}+y_{i+1}}{2} & i \text{ odd and } i \neq 2N-1 \\ \frac{y_{2N-2}}{2} & i = 2N-1 \end{cases} \tag{4.18}$$

RE, the reconstruction error, is defined as follows:

$$\begin{aligned} RE &= \sum_{i=0}^{2N-1} (x_i - \hat{y}_i)^2 \\ &= \sum_{n=0}^{N-1} (x_{2n} - y_{2n})^2 + \sum_{n=0}^{N-2} \left(x_{2n+1} - \frac{y_{2n}+y_{2n+2}}{2}\right)^2 + \left(x_{2N-1} - \frac{y_{2N-2}}{2}\right)^2 \end{aligned} \tag{4.19}$$

To minimize RE, we need to compute y_i to satisfy the following equations, for any even i.

$$\frac{\partial RE}{\partial y_i} = 0, \quad i = 0, 2, \cdots, 2N-2. \tag{4.20}$$

After substituting RE in Equation (4.19) into Equation (4.20) and simplifying the expression, we get the following matrix transformation:

$$\begin{pmatrix} y_0 \\ y_2 \\ \vdots \\ y_{2N-4} \\ y_{2N-2} \end{pmatrix} = \mathbf{T} \times \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{2N-2} \\ x_{2N-1} \end{pmatrix} \tag{4.21}$$

where

$$\mathbf{T} = \mathbf{A}^{-1} \times \mathbf{B} = \begin{pmatrix} 1 & \frac{1}{5} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \frac{4}{5} & \frac{2}{5} & & & \cdots & & & & & \\ 0 & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \cdots & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & \cdots & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & & \\ & & & & \cdots & & & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \end{pmatrix} \tag{4.22}$$

The condition for Equation (4.21) to have one and only one solution is when matrix **A** in Equation (4.22) is invertible. To prove this we only need to show the determinant of **A** to be nonzero, which is the necessary and sufficient condition for a matrix to be invertible. The determinant of **A** is shown as follows:

$$det(\mathbf{A}) = \begin{vmatrix} 1 & \frac{1}{5} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & 1 \end{vmatrix}_N \tag{4.23}$$

First, by applying Laplace expansion to the first line of **A**, we have

$$det(\mathbf{A}) = \begin{vmatrix} 1 & \frac{1}{6} & & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & 1 \end{vmatrix}_{(N-1)} - \frac{1}{5} \begin{vmatrix} \frac{1}{6} & \frac{1}{6} & & \cdots & 0 & 0 & 0 \\ 0 & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & 1 \end{vmatrix}_{(N-1)}. \tag{4.24}$$

After applying Laplace expansion to the two matrices on the right-hand side of Equation (4.24), we get

$$det(\mathbf{A}) = \begin{vmatrix} 1 & \frac{1}{6} & & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & 1 \end{vmatrix}_{(N-1)} - \frac{1}{30} \begin{vmatrix} 1 & \frac{1}{6} & & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & 1 \end{vmatrix}_{(N-2)}. \tag{4.25}$$

All the determinants now are for tridiagonal matrix $\mathbf{T}$ and can be calculated using:

$$det(\mathbf{T}_N) = \frac{\beta^{N+1} - \alpha^{N+1}}{\beta - \alpha} \tag{4.26}$$

where α and β are roots of equation $x^2 - x - (\frac{1}{6})^2 = 0$.

After substituting Equation (4.26) into Equation (4.25), it is easy to show that $det(\mathbf{A})$ is always nonzero, which proves that there is a unique $\vec{y}_{even}$ satisfying Equation (4.21).

From Equation (4.21) we obtain y_{2n}, $n = 0, 1, \cdots, N-1$. In order to get the complete transformation from $\vec{x}$ to $\vec{y}$, $\vec{y}_{odd} = y_1, y_3, \cdots, y_{2N-1}$ is needed, which can be obtained by following the same procedure for deriving $\vec{y}_{even}$.

In case that the packet containing all the even samples is lost, Equation (4.18) is changed to

$$\hat{y}_i = \begin{cases} y_i, & i \ odd \\ \frac{y_{i-1}+y_{i+1}}{2} & i \text{ even and } i \neq 0 \\ \frac{y_1}{2} & i = 0 \end{cases} \tag{4.27}$$

Also, RE defined in Equation (4.19) is changed to:

$$\begin{aligned} RE &= \sum_{i=0}^{2N-1} (x_i - \hat{y}_i)^2 \\ &= \sum_{n=0}^{N-1} (x_{2n+1} - y_{2n+1})^2 + \sum_{n=1}^{N-1} \left(x_{2n} - \frac{y_{2n-1}+y_{2n+1}}{2}\right)^2 + \left(x_0 - \frac{y_1}{2}\right)^2 \end{aligned} \tag{4.28}$$

Finally, the transformation for $\vec{y}_{odd}$ is as follows:

$$\vec{y}_{odd} = \begin{pmatrix} 1 & \frac{1}{6} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & 1 & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{5} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & & \cdots & & & & & \\ & & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \cdots & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & \cdots & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \\ & & & & \cdots & & & & \frac{2}{5} & \frac{4}{5} \end{pmatrix} \vec{x} \tag{4.29}$$

Similarly, we can prove that Equation (4.29) will always have one and only one solution.

Now the transformation from $\vec{x}$ to $\vec{y}$ is complete. The sender can send the transformed signals over the Internet, and if the loss is isolated, the receiver can reconstruct the original signals near optimally. The reason why optimality is only "near" is that $\vec{y}$ takes floating point values after transformation and is generally cast to lower precision (like integers) to reduce the bandwidth in transmission. Luckily, the loss of precision has minor effect on the reconstruction quality since only an average rounding error of ± 0.5 per sample is introduced.

Figure 1.2(b) illustrates the result of applying the transformation procedure presented in this subsection on a segment of voice samples. The reconstruction error based on the transformed even samples and reconstructed odd samples is 4.03 dB (Figure 1.2(b)), whereas the reconstruction error based on the original even samples and reconstructed odd samples is 1.69 dB (Figure 1.2(a)).

Case II: No packet loss in an interleaving pair

If both packets in an interleaving pair are received, we can obtain the following inverse transformation to restore $\vec{x}$ by combining Equations (4.21) and (4.29):

$$\vec{x} = \begin{pmatrix} \frac{4}{5} & \frac{2}{5} & & & & & & & & \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & & & & & & & \\ 0 & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \\ & & & & & & & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \\ & & & & & & & & \frac{2}{5} & \frac{4}{5} \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & \frac{1}{5} & 0 & & & & & & & \\ 0 & 1 & 0 & \frac{1}{6} & & & & & & & \\ \frac{1}{6} & 0 & 1 & 0 & \frac{1}{6} & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & & \frac{1}{6} & 0 & 1 & 0 & \frac{1}{6} \\ & & & & & & & \frac{1}{6} & 0 & 1 & 0 \\ & & & & & & & & \frac{1}{5} & 0 & 1 \end{pmatrix} \vec{y} \tag{4.30}$$

After computing $\vec{x}$ using Equation (4.30) at the receiver, the original stream $\vec{x}$ can be restored if $\vec{y}$ does not have any precision loss during processing and transmission. The proof for the existence and uniqueness of the solution is similar to that in the last subsection.

If there is precision loss when we cast $\vec{y}$ from floating-point values to 16-bit signed integers, then the reconstruction quality may obviously be not as good as that using floating point values. The reason is that the inverse transformation defined in Equation (4.30) is actually a linear transformation, and the loss of precision in every component of $\vec{y}$ will be accumulated. Fortunately, by carefully choosing the size of the transformation matrices, the original signals can be reconstructed quite well. Section 4.2.4 shows that the SNR of the reconstructed signal can reach around 30 dB.

In summary, our method always performs better than the method without transformation in the presence of loss. It gives near optimal reconstruction if one packet of an interleaving pair is lost, and can reconstruct the original signals if no packet is lost.

4.2.2 Coping with longer bursty losses of packets

From the results of Chapter 2, we know that domestic sites generally have bursty losses of one packet in duration, whereas international sites may have bursty losses of three or more packets. There are three ways to cope with longer bursty losses.

1. If the two packets in an interleaving pair are not transmitted consecutively but are separated by M packets from other interleaving pairs, then the arrangement can accommodate a bursty loss of M packets, but may not be able to accommodate cases when both packets in an interleaving pair are lost due to packet losses of burst length one. For example, let M be 4 and packets $P_{i,a}$ and $P_{i,b}$ be the two packets of interleaving pair i. Then a bursty loss of one packet can be tolerated if four interleaving pairs are transmitted in the following order: $P_{1,a}, P_{1,b}, P_{2,a}, P_{2,b}, P_{3,a}, P_{3,b}, P_{4,a}, P_{4,b}$; whereas a bursty loss of four packets can be tolerated if the four pairs are transmitted in the following order: $P_{1,a}, P_{2,a}, P_{3,a}, P_{4,a}, P_{1,b}, P_{2,b}, P_{3,b}, P_{4,b}$. However, the latter case cannot tolerate the loss of $P_{1,a}$ and $P_{1,b}$ due to two packet losses of burst length one. Another disadvantage of this scheme is that it increases the end-to-end delay because the receiver has to wait a longer time in order to assemble both packets in an interleaving pair. The distance to separate the two packets in an interleaving pair, M, is driven by feedback information from the receiver.

2. We can use larger interleaving factors to overcome packet losses. In general, M-way interleaving is able to overcome bursty packet losses of length $M-1$. (We call the related packets in M-way interleaving an *interleaving set.*) By extending the derivation in Section 4.2.1, we obtain the optimal transformation in order to overcome the loss of $M-1$ out of M interleaved packets. Without loss of generality, let samples $y_0, y_M, y_{2M}, \cdots$ be the only signals received. A general

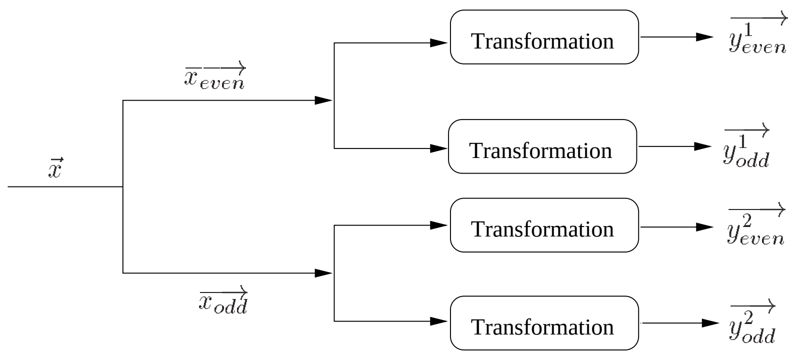

$\overrightarrow{x}$
$\overrightarrow{x_{even}}$
$\overrightarrow{x_{odd}}$
Transformation
Transformation
Transformation
Transformation
$\overrightarrow{y^1_{even}}$
$\overrightarrow{y^1_{odd}}$
$\overrightarrow{y^2_{even}}$
$\overrightarrow{y^2_{odd}}$

Figure 4.7: Constructing four-way interleaving using two two-way interleaving.

interpolation-based reconstruction method can then be written as

$$\hat{y}_i = \alpha_i y_{kM} + (1-\alpha_i) y_{(k+1)M}, \quad \forall i,\ kM < i < (k+1)M. \tag{4.31}$$

By minimizing the reconstruction error with respect to $y_0, y_M, y_{2M}, \cdots$, we obtain one of the interleaved streams $y_0, y_M, y_{2M}, \cdots$. By following the same procedure, we can derive the other $M-1$ streams using the optimal transformation.

Note that the optimal transformation for each interleaved stream is based on the assumption that all the other $M-1$ streams are lost; hence, reconstructions is suboptimal when less than $M-1$ streams are lost. The disadvantage of this scheme is that in the worst case, $M-1$ consecutive samples will be lost and have to be reconstructed using the two samples received around the block. Such reconstruction may be less accurate as compared to the first alternative. Also, it is difficult to guarantee the reconstruction quality for cases in which less than $M-1$ packets in an interleaving set were lost.

3. We can construct M-way interleaving using a combination of 2-way interleaving. For convenience, we use 4-way interleaving as an example.

Figure 4.7 shows the four streams created at the sender. The sender first interleaves the original stream $\vec{x}$ to even ($\overrightarrow{x_{even}} = x_0, x_2, \cdots$.) and odd ($\overrightarrow{x_{odd}} = x_1, x_3, \cdots$.) streams. Second, it transforms $\overrightarrow{x_{even}}$ and $\overrightarrow{x_{odd}}$ to get four streams $\overrightarrow{y^1_{even}}$, $\overrightarrow{y^1_{odd}}$, $\overrightarrow{y^2_{even}}$, and $\overrightarrow{y^2_{odd}}$, using the transformation procedure discussed in Section 4.2.1.

At the receiver, we can construct $\overrightarrow{x_{even}}$ when there is no loss, by deinterleaving $\overrightarrow{y^1_{even}}$ and $\overrightarrow{y^1_{odd}}$ and by applying inverse transformation on the de-interleaved stream. Similarly, we can reconstruct $\overrightarrow{x_{odd}}$ after deinterleaving $\overrightarrow{y^2_{even}}$ and $\overrightarrow{y^2_{odd}}$ and by inverse transformation of the deinterleaved stream.

At the receiver, there are four possibilities of loss on the packets received. In the first case, three packets in an interleaving set were lost. For example, assume $\overrightarrow{y^1_{odd}}$, $\overrightarrow{y^2_{even}}$, and $\overrightarrow{y^2_{odd}}$ were lost. By our explanation in Section 4.2.1, we can optimally reconstruct $\overrightarrow{y^1_{odd}}$ from $\overrightarrow{y^1_{even}}$ received. Since $\overrightarrow{y_1}$ is now reconstructed, we can reconstruct each sample in $\overrightarrow{x_{odd}}$ by computing the average of its adjacent even samples. In this case, the reconstruction quality is guaranteed to be better than that without transformation, because $\overrightarrow{y^1_{odd}}$ is optimally reconstructed.

In the second case, two packets that are both in either $\overrightarrow{y^1}$ or $\overrightarrow{y^2}$ were lost. For instance, assume $\overrightarrow{y^2_{even}}$ and $\overrightarrow{y^2_{odd}}$ were lost. We can apply the inverse transformation discussed in *Case II* in Section 4.2.1 on $\overrightarrow{y^1_{even}}$ and $\overrightarrow{y^1_{odd}}$ to obtain $\overrightarrow{x_{even}}$, and recover $\overrightarrow{x_{odd}}$ by averaging. The reconstruction quality may not always be better than the case without transformations because $\overrightarrow{x_{even}}$ was not optimized for the reconstruction of $\overrightarrow{x_{odd}}$.

In the third case, two packets, one in $\overrightarrow{y^1}$ and one in $\overrightarrow{y^2}$, were lost. Using our method, both lost packets can be optimally reconstructed based on the packets received. Hence, we can guarantee better performance than the case without transformations.

In the last case, only one packet in an interleaving set was lost. For example, assume $\overrightarrow{y^2_{odd}}$ was lost. Using our method, we can optimally reconstruct $\overrightarrow{y^2_{odd}}$ from $\overrightarrow{y^2_{even}}$ received. We can further reconstruct $\overrightarrow{x_{even}}$ by applying our inverse transformation on $\overrightarrow{y^1_{even}}$ and $\overrightarrow{y^1_{odd}}$. Without any loss of computational precision, we can also guarantee better reconstruction quality than the case without transformations.

In summary, the last method for coping with longer bursty losses is better than the first method because it can tolerate bursty losses of a maximum of three consecutive packets. It is also better than the second method because we can guarantee its reconstruction quality to be better than the case without transformation.

Obviously, a large M is able to tolerate more consecutive packet losses, at the expense of longer delays, because the receiver will have to assemble all the interleaved streams before playing the combined stream. On the other hand, a small M cannot overcome many packet losses, but has shorter delay that is sometimes critical in real-time applications. A suitable M for real-time audio transmissions by the sender should be driven by feedback information obtained from the receiver.

The three methods proposed here are equivalent in terms of the additional amount of end-to-end delay added to the transmission. This is true when a given segment of voice samples are structured either using 2-way interleaving and consecutive packets in an interleaving pair are separated by

$M-1$ packets from other interleaving pairs, or using M way interleaving and consecutive packets in the interleaving set are sent one after the other.

4.2.3 Computational complexity of two-way interleaving

The computational complexity of our proposed transformation is low, as all the matrices as well as their inverses can be computed beforehand if their dimensions are known. The actual online computation involves only a few multiplications and additions.

To illustrate the computational complexity for the sender, we use the transformation in Equation (4.21) for two-way interleaving as an example. Substituting $\mathbf{T}$ in Equation (4.21) with $\mathbf{A}^{-1}\mathbf{B}$, we have $\vec{y} = \mathbf{A}^{-1} \times (\mathbf{B}\vec{x})$. If every element of $\mathbf{A}$ is nonzero, then the computational cost for $\mathbf{B}\vec{x}$ is $3N$ multiplications plus $2N$ additions for every N samples. To get y, the computational cost is N^2 multiplications plus $(N-1) \times N$ additions. The overall complexity for solving the equations is, therefore, $(N+3) \times N$ multiplications and $(N+1) \times N$ additions. In other words, the complexity for each sample is $N+3$ multiplications and $N+1$ additions.

In practice, the complexity for large N is far less than the above because not all entries of $\mathbf{A}^{-1}$ are effective in calculating $\vec{y}$. A simple explanation is given as follows.

Assume that the voice data for processing is in 16-bit linear PCM wave format in the range $[-32768, 32767]$, and that a ± 1 change of sample value within the range will not be perceptible to human ears. We observe that the maximum absolute value in vector $\vec{x'}$ ($\vec{x'} = \mathbf{B}\vec{x}$) is less than $M = \frac{4}{3} \times 32768$ because the summation of any row of matrix $\mathbf{B}$ is less than $\frac{4}{3}$. A transformed signal y_{2n} can be computed using

$$y_{2n} = \lfloor \sum_{k=1}^{N} (\mathbf{A}^{-1}_{n,k} \times x'_k) \rfloor \tag{4.32}$$

If $|\mathbf{A}^{-1}_{n,k}| < \frac{1}{2MN}$, then the difference with or without $\mathbf{A}^{-1}_{n,k} \times x'_k$ is at most $\frac{1}{2N}$. Thus, $\sum(\mathbf{A}^{-1}_{n,k} \times x'_k)$ is less than 0.5 for all $|\mathbf{A}^{-1}_{n,k}| < \frac{1}{2MN}$. Based on our assumption, all the items in the summation can be neglected. Hence, only those elements in matrix $\mathbf{A}^{-1}$ whose value is greater than $\frac{1}{2MN}$ are effective. For other elements, we can simply set them to zero. Obviously, the computational cost can be greatly reduced if such elements occur frequently in the matrix. After calculating $\mathbf{A}^{-1}$, we found that only about 17 items out of each line are larger than $\frac{1}{2MN}$ for $N > 17$. The total complexity is therefore reduced to $3 + 17 = 20$ multiplications and a few additions for each sample when N is greater than 17.

The computational complexity at the receiver needs to be analyzed separately for the two cases presented in Section 4.2.1. For *Case I*, the computational cost at the receiver is low because it only fills in the missing samples by taking the average of adjacent samples received. For *Case II*, each sample will need $2N$ multiplications and $2N-1$ additions for the inverse transformation in Equation (4.30). Our experimental results in the next section show that N is generally small, between 32 and 64; hence, the computational cost per sample is low.

4.2.4 Experimental results

In this section, we apply our transformation algorithms to the same four audio files described in the beginning of this chapter.

The first set of experiments is for *Case I* in Section 4.2.1 in which only one packet in an interleaving pair was lost. For simplicity, we only consider the case when all the odd samples were lost. The performance comparisons between the method we present in Section 4.2.1 and the case without transformation are shown in Figure 4.8. (The results are quite similar for the case when all the even samples were lost.) For every voice file, we tried different sizes of the transformation matrix: 4, 8, 16, 32, 64, 128, and 256. Figures 4.8(a)-4.8(c) compare our algorithms when applied to voice typed audio files, whereas Figure 4.8(d) compares our algorithms when applied to a music file.

The results show that our transformation-based reconstruction method is applicable to both voice and music files and improves over the case when no transformation is performed. Moreover, in case of loss, the reconstruction method based on transformed samples is consistently better than the method without transformation by about 1 to 2 dB, which means the reconstruction error of the method based on transformation is as low as $79\% - 63\%$ of the reconstruction error introduced by the method without transformation.

Another observation on the results is that N does not have any significant effect on the reconstruction quality when $N > 64$. Hence, there is no need to use very large matrices to do the transformation. This is important in our implementation because it takes less space to store smaller transformation matrices. Moreover, we show in the next set of experiments that a smaller transformation matrix achieves less reconstruction error when no packets are lost.

The absolute SNR values differ in these figures due to several reasons. Both the voice volume and quality can affect the absolute value of SNR. For example, if the voice quality is poor and the signals contain a lot of random and discontinuous noise, then it is hard for the missing samples to

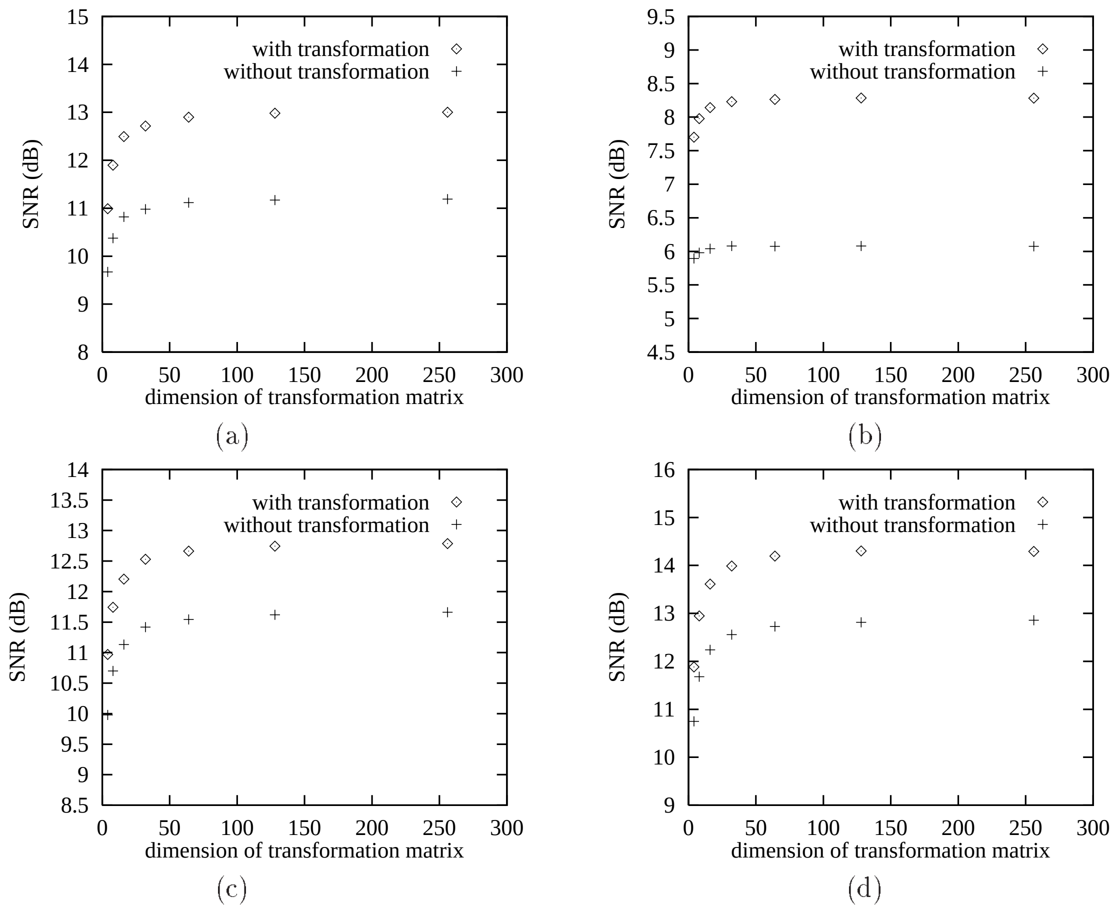


Figure 4.8: Comparison of reconstruction quality between using transformation versus no transformation on (a) a movie clip of a talk, (b) a movie clip of a man's voice, (c) a segment of woman's speech, and (d) a segment of music.

Table 4.3: The quality of reconstructed information based on transformed voice samples for two-way interleaving. N is the size of the transformation matrix. *Loss* represents the case in which one of the two interleaved streams was lost. *No Loss* represents the case in which both streams were received.

Sound	*SNR* (dB)									
	$N = 32$		$N = 40$		$N = 48$		$N = 56$		$N = 64$	
File	Loss	No Loss	Loss	No Loss	Loss	No Loss	Loss	No Loss	Loss	No Loss
1	12.71	28.37	12.82	25.19	12.85	23.19	12.83	21.02	12.90	19.14
2	8.23	31.09	8.23	28.11	8.25	26.12	8.25	23.96	8.26	22.13
3	12.57	32.12	12.57	28.87	12.63	26.47	12.71	24.39	12.70	23.25
4	13.99	34.56	14.06	31.54	14.12	29.02	14.18	27.31	14.19	25.62

be reconstructed using the average of neighboring samples. Also, if the voice volume is low, then noise may mislead the reconstruction procedure.

The second set of experiments is for *Case II* in Section 4.2.1 when both packets in an interleaving pair are received by the receiver. Table 4.3 shows the reconstruction quality for different sizes of the transformation matrix. The largest size of the transformation matrix chosen is 64, at which point near optimal reconstruction quality can be achieved. For every size of the transformation matrix, we compute the SNRs for the cases with and without loss.

The table shows that the SNRs for cases of no loss are consistently decreasing when the size of the transformation matrix is increasing, whereas the SNRs for cases with loss are consistently increasing. The reason for this behavior is that for a given N, $2N$ simultaneous equations are to be solved in order to get $\vec{x} = x_0, x_1, \cdots, x_{2N-1}$ from $\vec{y} = y_0, y_1, \cdots, y_{2N-1}$ using (4.30). This causes each x_i, $i \in [0,\ 2N-1]$, to be related to all the other $2N-1$ values, and numerical errors in the process of solving (4.30) when some y_i has rounding errors will be accumulated. Our results show that a larger N will introduce more rounding errors and worsen reconstruction quality.

Another observation from Table 4.3 is that the degradation in reconstruction quality is only around 0.2 dB for the case of loss when N changes from 64 to 32, whereas the improvement in reconstruction quality for the case of no loss is over 9 dB. When SNR reaches around 30 dB at $N = 32$, the reconstruction quality is usually satisfactory.

The choice of the proper value of N may also depend on the loss behavior of the connection. If the connection has very few losses, then smaller N's are preferred, whereas if losses are frequent, then larger N's are better.

Finally, Table 4.4 shows the reconstruction quality for 4-way interleaving based on two 2-way interleaving. In the first three cases in which two or three packets in an interleaving set were lost, our proposed transformation method has better or equal reconstruction quality as compared to the

Table 4.4: Reconstruction quality for four-way interleaving based on three two-way interleaving for cases with and without transformation. **I** represents the case in which three out of four interleaved streams were lost. **II** represents the case in which one of the two groups of the four interleaved streams was lost. **III** represents the case in which two streams, each in a different group, were lost. **IV** represents the case in which one stream out of four interleaved streams was lost.

	SNR (dB)							
Sound	**I**		**II**		**III**		**IV**	
File	With	Without	With	Without	With	Without	With	Without
1	7.93	6.68	11.19	11.23	9.17	7.77	13.06	14.28
2	4.55	4.09	6.08	6.08	8.05	6.90	11.20	9.10
3	7.37	6.18	11.66	11.68	8.25	6.74	12.99	14.60
4	9.39	8.28	12.89	12.90	10.53	9.12	14.63	15.92

case without transformation. The reconstruction quality in the last case is worse due to precision loss. However, this is a case that should not happen frequently when 4-way interleaving is applied and when we experience a high percentage of long bursty losses.

4.3 System Test

The experimental results shown in Sections 4.1 and 4.2 do not reflect realistic reconstruction quality because the loss patterns were created synthetically, and the experiments did not consider the influence of compression on the reconstruction quality. We need to test our algorithms on a real network using the performance measures we have defined (SNR, end-to-end delays, and jitters). In this section, we test the two reconstruction methods proposed on the Internet via a real-time transmission system we have built.

The tests were done solely on our local computers for ease of measuring end-to-end delays and jitters. They are three processes running simultaneously on a local computer: the sender process, the echo-daemon process, and the receiver process. The sender process records voice from a microphone, time-stamps the data, interleaves the samples, processes the data if the samples are to be transformed, compresses the voice, and sends the data to the echo-daemon at the local computer. The echo-daemon process is responsible for sending the data to the echo port of a remote machine, and for forwarding the data received to the receiver process. The receiver process receives the data from the echo daemon, decompresses, deinterleaves or reconstructs in case of loss, notes the playback time, and saves the reconstructed voice stream to the disk for future computation of SNR. In our tests, we chose two remote hosts among the six hosts listed in Chapter 2: Stanford and China, each representing quite different traffic patterns. Due to the use of the echo port, the end-to-end delays measured in these tests were longer than the corresponding actual one-way connections. Performance measures (SNR, end-to-end delay and jitter) of the system were computed once an hour for 24 hours. In our implementation of transformation-based reconstruction method, we did not perform any inverse transformation when both packets in an interleaving pair were received, because compression introduces more significant noise in the signals than precision loss discussed in the last section, and an inverse transformation of noisy inputs will cause significant errors.

The first test compares the reconstruction quality measured in SNR for the two schemes discussed in Sections 4.1 and 4.2 with respect to the reconstruction method based on simple averaging. Figure 4.9(a) shows the improvement in SNR for the Stanford-Champaign connection. The average SNR is 13.10 dB if reconstruction is based on simple averaging alone, and the average SNR is 13.30 dB if reconstruction is based on transformation and simple averaging. Figure 4.10(a) shows the improvement in SNR for the China-Champaign connection. The average SNR is 9.27 dB if reconstruction is based on simple averaging alone, and the average SNR is 10.10 dB if reconstruction is based on transformation and simple averaging. Both figures show better performance for the transformation-based reconstruction method. It is not surprising that the improvements are greater when the loss rate is higher because the goal of transformation-based reconstruction is to improve reconstruction quality in case of loss, but transformation may introduce some noise in case of no loss.

Figures 4.9(b) and 4.10(b) show the results corresponding to the case using adaptive filtering. Figure 4.9(b) compares simple averaging (with average SNR of 13.14 dB) with adaptive filtering (with average SNR of 13.08 dB) for the Stanford-Champaign connection, whereas Figure 4.10(b) compares simple averaging (with average SNR of 8.10 dB) with adaptive filtering (with average SNR of 8.14 dB) for the China-Champaign connection. Both graphs show little improvement for adaptive filtering. The mediocre performance of adaptive filtering is due to difficulties in choosing suitable values of the filter parameters to fit the characteristics of the voice stream.

Figures 4.9(c) and 4.9(d) compare the average end-to-end delays and jitters over a 24-hour period between simple averaging and reconstruction based on transformation for the Stanford-Champaign connection. The average end-to-end delay and jitter for simple averaging are 215.56 ms and 72.36 ms, respectively, whereas the average end-to-end delay and jitter with transformation are 238.34 ms and 75.94 ms, respectively. In our experiments, because every packet covers a sampling period of 32 ms with an interleaving factor of 2, the playing jitter should theoretically be 64 ms. In practice, due to scheduling overhead in the operating system, overhead in the sound-card driver buffer management, and processing time, jitters of around $70 - 80$ ms are pretty reasonable.

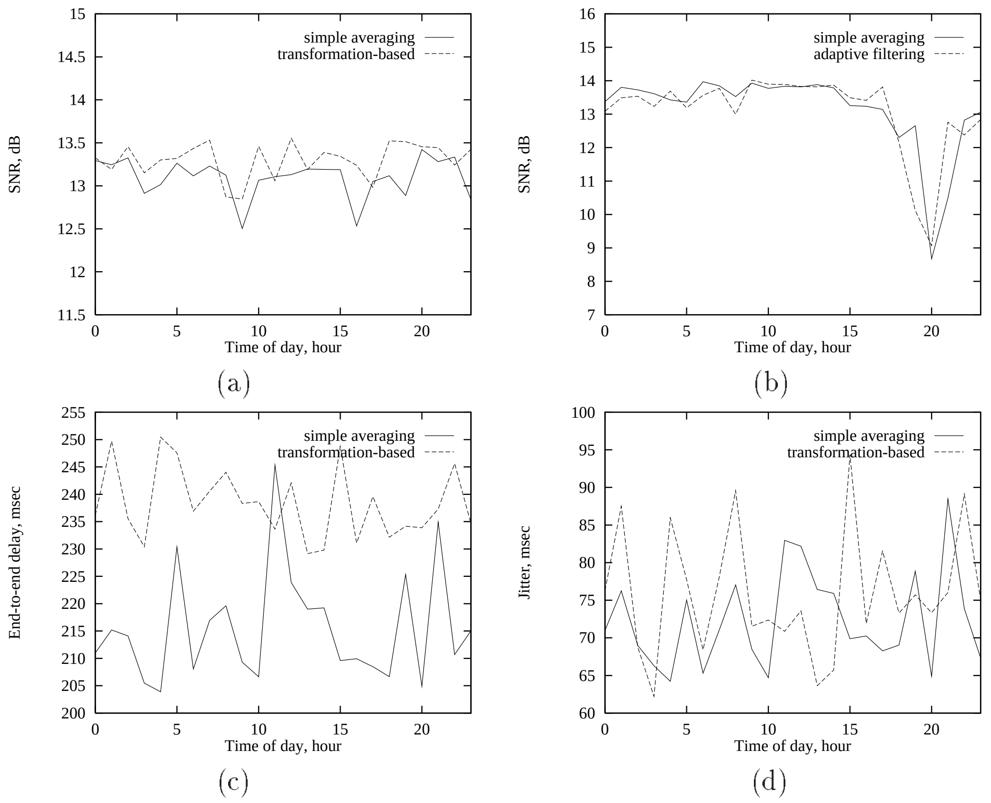


Figure 4.9: Performance of the proposed reconstruction methods for the Stanford-Champaign connection: (a) comparison of quality between reconstruction based on simple averaging and reconstruction based on transformation, (b) comparison of quality between reconstruction based on simple averaging and reconstruction based on adaptive filtering, (c) comparison of end-to-end delay between reconstruction with transformation and simple averaging, and (d) comparison of jitters between reconstruction with transformation and simple averaging.

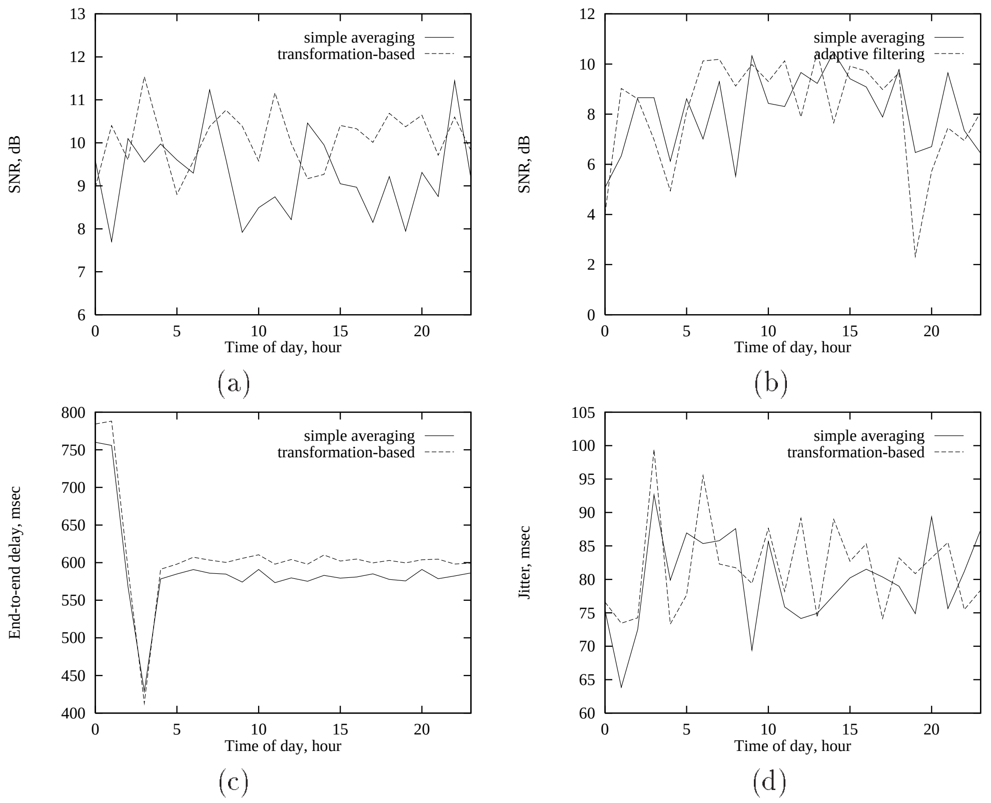


Figure 4.10: Performance of the proposed reconstruction methods for the China-Champaign connection: (a) comparison of quality between reconstruction based on simple averaging and reconstruction based on transformation, (b) comparison of quality between reconstruction based on simple averaging and reconstruction based on adaptive filtering, (c) comparison of end-to-end delay between reconstruction with transformation and simple averaging, and (d) comparison of jitters between reconstruction with transformation and simple averaging.

Figures 4.10(c) and 4.10(d) compare the performance for the China-Champaign connection. The average end-to-end delay and jitter for simple averaging are 589.62 ms and 79.88 ms, respectively, whereas the average end-to-end delay and jitter for reconstruction based on transformation are 609.13 ms and 81.70 ms, respectively.

For both connections, jitters are almost the same with and without transformations. The extra end-to-end delay introduced by transformation is less than 40 ms and is tolerable. Last, the end-to-end delay requirement of 300 ms can be satisfied for domestic connections, whereas it is hard to be satisfied for international connections.

In summary, our tests over the Internet show that our system and the transformation-based reconstruction method work well.

4.4 Summary

This chapter describes two new reconstruction methods built on top of simple averaging. The method based on adaptive filtering focuses on modifying the average reconstructed signals at the receiver. It can improve the reconstruction quality for voice signals once suitable parameters, such as filter length and adaptation-step size, are known. Moreover, it needs time to adapt to the time-varying characteristics of the voice signals. On the other hand, the method based on transformed inputs allows the sender perform a transformation in order to facilitate reconstruction at the receiver. Further, it can be easily adapted to interleaving factors other than 2.

We have tested both methods over the Internet and have compared the performance in terms of SNR, end-to-end delay, and jitter with respect to simple averaging. Our results show that reconstruction based on adaptive filtering gives no improvement over simple averaging, whereas reconstruction based on transformation can improve SNR, while keeping the end-to-end delay low.

CHAPTER 5

CONCLUSIONS

This thesis investigates the necessary components of a high-quality real-time voice transmission system, and presents new reconstruction algorithms that can achieve better reconstruction quality.

Reliable transmissions of real-time voice streams over the Internet are difficult because no underlying protocol supports transfers with given QoS requirements. TCP is not realistic to deal with delay-sensitive multimedia data. By using UDP, a connectionless, best-effort transport-layer protocol, applications need to handle out-of-order packets, loss, long delay, and large jitter, that are inherent in the resource-limited Internet.

Voice traffic has unique characteristics. It is regular, has relatively smaller packet size and low-bandwidth requirement as compared to video traffic. Experiments show that voice-traffic behavior is time varying and destination dependent. Also, a voice stream may encounter large jitters and high loss, while the length of consecutive packet losses is rather small (usually 1-2). These observations make interleaving and reconstruction both practical and attractive. It is practical because interleaving will not introduce too much delay, usually only one more packet delay at each end. It is attractive because it does not need to send redundant information to overcome loss.

To handle all the above problems, we examine the necessary components of the delivery system. To reduce bandwidth usage, both silence removal and compression are important, although both will introduce more delay. The compression scheme chosen for a software-oriented system needs to have low complexity and low delay. The G.727 scheme is usually preferred over G.723.1 in this case mainly because of its simplicity. After taking silence suppression into account, the bandwidth required by G.723.1 is 5.3 kb/s, whereas the bandwidth required by G.727 is around 8 kb/s. To adapt the system to changes in the network, a statistics collector is used to calculate loss, delay, jitter, etc., and feeds the results to different system components. To handle out-of-order packets and large delay variance, jitter buffers are included whose size is chosen based on estimated average delay and delay variance. Introducing jitter buffers will remove discontinuity in the receiving voice stream while increasing end-to-end delay. The size of the jitter buffers is a critical parameter to balance between delay and quality.

In this thesis, we have proposed new schemes for handling loss via interleaving and reconstruction. We have investigated two methods to enhance the reconstructed signals. Existing reconstruction methods that recover missing signals based on the average of adjacent samples are quite simple and generally work well. However, they do not take advantage of any signal-specific information and have difficulty in dealing with rapidly changing voice signals.

One new method tested applies adaptive filtering on top of the average reconstruction. Adaptive filtering is able to explore trends in the signal. To improve adaptive filtering, important parameters, such as filter length and adaptation step size, need to be chosen carefully for a specific voice stream. Since this is generally not possible to realize in real time, universal parameters are tested for different voice streams. This leads to poor performance because the method has difficulty in dealing with rapidly changing parts of voice signals.

A new idea to overcome this difficulty is to smooth the original signals before sending them out. This idea leads to the transformation-based reconstruction method that allows the sender to transform the original signals in order to facilitate better reconstruction at the receiver. This method has two advantages: it performs well and it is easy to adjust.

We have tested both methods over the Internet and have measured the transmission quality based on SNR, end-to-end delay, and jitter. We have found that adaptive filtering improves very little on the average, whereas the transformation-based reconstruction can improve SNR, especially for connections with heavy losses. The new method also incurs additional but tolerable end-to-end delay (below 40 ms) and no additional jitter (since both use the same jitter-buffer strategy).

In conclusion, our new transformation-based reconstruction algorithm can improve over the reconstruction method based on simple averaging for transmitting voice over the current Internet.

REFERENCES

[1] L. Sweet, "Toss your dimes-Internet video phones have arrived," *ZD Internet Magazine*, August 1997, http://www.zdnet.com/products/content/zdim/0208/zdim0010.html.

[2] M. S. Shuster, "Diffusion of network innovation: implications for adoption of Internet services," M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1998.

[3] S. Bass, "Internet phones take on Ma Bell," *PC World*, June 1997, http://www.pcworld.com/software/internet_www/articles/jun97/1506p165.html?SRC=mag.

[4] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the 13th Annual Joint Conference of the IEEE Computer and Communications Societies on Networking for Global Commmunication*, vol. 2, 1994, pp. 680–688.

[5] J. C. Bolot and A. V. Garcia, "Control mechanisms for packet audio in the Internet," in *Proceedings of IEEE INFOCOM*, April 1996, pp. 232–239.

[6] Z. G. Chen, "Coding and transmission of digital video on the Internet," Ph.D. dissertation, University of Illinois, Urbana-Champaign, IL, 1997.

[7] B. Dempsey, T. Strayer, and A. Weaver, "Adaptive error control for multimedia data transfers," in *Proceedings of International Workshop on Advanced Communications and Applications for High Speed Networks*, March 1992, pp. 279–289.

[8] G. Held, *Voice Over Data Networks.* New York: McGraw-Hill, 1998.

[9] T. J. Kostas, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, and J. Mahler, "Real-time voice over packet-switched networks," *IEEE Network*, vol. 12, pp. 18–27, January-February 1998.

[10] B. J. Dempsey and Y. Zhang, "Destination buffering for low-bandwidth audio transmission using redundancy-based error control," in *Proceedings of 21st IEEE Local Computer Networks Conference*, October 1996, pp. 345–355.

[11] V. Hardman, M. A. Sasse, M. Handley, and A. Watson, "Reliable audio for use over the Internet," in *International Networking Conference*, June 1995, pp. 171–178.

[12] Telogy Networks, "Voice over packet tutorial," November 1997, http://www.webproforum.com/telogy/full.html.

[13] N. Shacham and P. McKenney, "Packet recovery in high-speed networks using coding and buffer management," in *Proceedings of IEEE INFOCOM*, May 1990, pp. 124–131.

[14] J. C. Bolot and P. Hoschka, "Adaptive error control for packet video in the Internet," in *International Conference on Image Processing*, vol. 1, September 1996, pp. 25–28.

[15] N. S. Jayant and S. W. Christensen, "Effects of packet losses in waveform coded speech and improvements due to odd-even sample-interpolation procedure," *IEEE Transactions on Communications*, vol. 29, pp. 101–110, February 1981.

[16] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 functional specification," Internet Requests for Comments 2205, September 1997, http://www.cis.ohio-state.edu/htbin/rfc/rfc2205.html.

[17] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real time applications," Internet Requests for Comments 1889, January 1996, http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1889.txt.

[18] D. E. Comer, *Internetworking with TCP/IP*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[19] P. G. Drago, A. M. Molinari, and F. C. Vagliani, "Digital dynamic speech detectors," *IEEE Transactions on Communications*, vol. 26, pp. 140–145, January 1978.

[20] J. F. Lynch, L. J. Josenhans, and R. E. Crochiere, "Speech/silence segmentation for real-time coding via rule based adaptive endpoint detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1987, pp. 1348–1351.

[21] S. Jacobs, A. Eleftheriadis, and D. Anastassiou, Columbia University, "Silence detection for multimedia communication," April 1997, http://www.ee.columbia.edu/~eleft/papers/mmsj97.html.

[22] M. H. Savoji, "A robust algorithm for accurate endpointing of speech signals," *Speech Communication (Netherlands)*, vol. 8, pp. 45–60, March 1989.

[23] C. Loo and R. W. Donaldson, "An adaptive silence deletion algorithm for compression of telephone speech," in *1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 2, 1997, pp. 701–705.

[24] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterance," *The Bell System Technical Journal*, vol. 54, pp. 297–315, February 1975.

[25] A. M. Kondoz, *Digital Speech: Coding for Low Bit Rate Communications Systems.* Chichester: John Wiley & Sons, 1994.

[26] D. Minoli and E. Minoli, *Delivering Voice over IP Networks.* New York: Wiley Computer Publishing, 1998.

[27] D. Kemp, R. A. Sueda, and T. E. Tremain, "An evaluation of 4800bps voice coders," in *1989 International Conference on Acoustics, Speech and Signal Processing*, May 1989, pp. 200–203.

[28] B. H. Juang, "The past, present, and future of speech processing," *IEEE Signal Processing Magazine*, vol. 15, pp. 24–48, May 1998.

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

[30] A. S. Spanias, "Speech coding: a tutorial review," in *Proceedings of the IEEE*, vol. 82, October 1994, pp. 1441–1582.

[31] F. Alvarez-Cuevas, M. Bertran, F. Oller, and J. M. Selga, "Voice synchronization in packet switching networks," *IEEE Networks*, vol. 7, pp. 20–25, September 1993.

[32] W. Montgomery, "Techniques for packet voice synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1022–1028, December 1983.

[33] W. E. Naylor and L. Kleinrock, "Stream traffic communication in packet switched networks: destination buffering constraints," *IEEE Computer*, vol. 30, pp. 2527–2534, December 1982.

[34] O. Macchi, *Adaptive Processing: The Least Mean Squares Approach with Applications in Transmission.* Chichester, England: John Wiley & Sons, 1995.

[35] C. Cowan and P. Grant, *Adaptive Filters.* Englewood Cliffs, NJ: Prentice-Hall, 1985.

[36] J. R. Treichler, C. R. Johnson, and M. G. Larimore, *Theory and Design of Adaptive Filters.* New York: John Wiley & Sons, 1987.

[37] B. Widrow and S. Stearns, *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1985.