RETRANSMISSION-BASED ERROR CONTROL FOR CONTINUOUS MEDIA TRAFFIC IN PACKET-SWITCHED NETWORKS

A Dissertation
Presented to
The Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment
of the Requirements for the Degree

Doctor of Philosophy
in
Computer Science

by

Bert J. Dempsey
May 1994

Approval Sheet

This dissertation is submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy (Computer Science)

Author's Name

This dissertation has been read and approved by the Examining Committee:

Alfred C. Weaver, Advisor

Jörg Liebeherr, Co-Advisor

William A. Wulf, Chairman

Sang H. Son

Stephen G. Strickland

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

May 1994

Acknowledgments

There are three people without whom I could not have completed this dissertation. Alf Weaver saw me through the entire process with a continuous stream of valuable advice, multidimensional support, and excellent opportunities. I especially appreciated the encouragement and support he offered during the tough times. Jorg Liebeherr spent uncountable hours with me, refining and improving the ideas in this dissertation. His clarity of thought and creativity is much evidenced in the final product. Finally, my wonderful wife, Molly, made the long process viable through her love and support at home.

The quality of the inhabitants of the Computer Networks Lab has been very high. Tim Hartrick provided me with many spirited discussions that have significantly broadened my understanding of computer science, major league sports, and Cleveland. John Fenton revealed to me the subtle workings of protocols and the importance of champagne. It was a pleasure to work with Matt Lucas on the empirical study given in Chapter 5 of the dissertation. Fraser Street and I had many fruitful conversations about the state-of-the-art. I also enjoyed and benefited from conversations with many others, especially Alex Colvin, James McNabb, Jeffrey Michel, Robert Simoncic, Mark Smith, and Alex Waterman.

I thank my friends outside the department—Lee Cohen, Jeff Herrin, Mark Langfitt, and Karen LeMaire—for their support and understanding during hectic times. I thank Louise Gallager for covering the acolytes and Boots Mead for enduring periods of limited technical support. Finally I thank my parents for their support and encouragement over the past 33 years—they have given me so much.

This work is supported in part by the Naval Command, Control, and Ocean Surveillance Center, Naval Research and Development Division, William T. Gex, Technical Representative.

Abstract

Distribution of digital audio and video, *continuous media*, over packet-switched networks has become increasingly feasible due to technology trends leading to powerful desktop computers and high-speed networks. Unlike reliable data transfers, transmission of continuous media streams is sensitive to network delays and has some tolerance for limited data loss. End-to-end protocols for continuous media traffic are now emerging, and an area of active research is error control in this context.

This dissertation provides a comprehensive and fundamental study of retransmission-based error control for the distribution of digital continuous media over packet-switched networks. While widely dismissed in the current literature, a retransmission-based approach is attractive since it imposes little overhead on network resources and since alternative techniques have notable drawbacks with respect to complexity, portability, and cost. It must be demonstrated, however, that retransmissions can provide significant error coverage while respecting delay constraints.

We define a novel delay-constrained retransmission scheme, Slack ARQ, and develop a simulation model to determine its feasibility for distribution of packet voice in a local area network (LAN). The evaluation uses a unique performance metric for retransmission effectiveness, incorporating both error and delay considerations that determine the overall transmission quality. This work is extended with an analytical end-to-end model for Slack ARQ, from which analytical expressions for our retransmission performance metric are derived. A principle reason for the paucity of retransmission-based approaches in the literature has been the lack of methodologies for assessing their effectiveness in a delay-sensitive environment. Our analysis provides, without resorting to lengthy simulations, quantification of the effects of transmission parameters, such as the packetization interval in the protocol and the network delay distribution, on retransmission success. Numerical examples show the applicability of Slack ARQ in many realistic transmission scenarios.

We design and carry out an empirical investigation of packet voice distribution across a contemporary high-performance campus-wide network. This study is of interest since large

multiple-segment LANs are likely candidates for near-term deployment of continuous media applications and since little empirical work has been done in this area. It is concluded that the campus-wide network in this study can indeed support real-time packet streams, but that sporadic high delays in the network may threaten transmission quality. We discuss the implications of the empirical measurements for our modeling, and, by calculating empirical probabilities, we show that the measurement data substantially corroborates the results of our simulation and analytical studies. That is, in total, the simulation, analysis, and empirical measurements presented in this dissertation conclusively demonstrate the feasibility of Slack ARQ in most packet-switched networks.

Finally, in Appendix A, we define a novel connection-oriented service that provides limited recovery from packet loss using delay-constrained retransmission within a next-generation transport protocol, the Xpress Transfer Protocol (XTP). We implement this lightweight service through minor modifications to an existing XTP implementation, and its performance is demonstrated in experimental network transfers.

.

To Molly

Contents

List of Tables

List of Figures

Chapter 1

Introduction

New application areas are emerging that will bring radical changes to the field of computer networking. High-speed fiber optic networks and increasingly powerful desktop computers are driving the trend toward a much higher degree of connectivity and the incorporation of new computing paradigms. Application domains such as wide-area distributed computing and digital multimedia—the integration of images, video, and audio data types with traditional text and graphics in digital computers—will create a new network traffic mix and introduce network flows with requirements quite different from those of current applications.

The clearest trend in this shift to a new mix of distributed applications is the rapidly growing emphasis on digital audio and video, or continuous media. [1] Digital representation of audio and video signals is fundamentally attractive since it offers more flexibility in manipulating and processing these data types than that available with analog representations. Integration of digital continuous media in general-purpose computing systems promises to significantly enhance the quality and bandwidth of human-computer interaction. As computing and communication merge, new digital communication services will be created.

Standards for digital representations are important for widespread acceptance. International standards for audio, images, and video have emerged in recent years, but standards continue to evolve as new techniques push the state-of-the-art in the coding and compression of audiovisual information [19]. At the present time (c. 1994) audio and video hardware is increasingly common in desktop computers, and the experience base for creating and using

[1] Continuous media refers to the fact that audio and video are fundamentally analog data types. Digitizing audio and video signals creates a bit stream that must be continuously transmitted across the network.

digital multimedia documents and interactive services is now expanding.

Continuous media services have traditionally been handled only in telephony and television networks that use analog methods and are separate from digital computer communications. With the convergence of enabling technology trends, the integration of digital continuous media and computer data into a single physical network has now become feasible, with potentially enormous economic benefits. Of particular importance is the highly visible effort by the international telephony standards body, CCITT, to standardize a high-speed integrated services network architecture known as Broadband ISDN, with its underlying packet-switched transport, the Asynchronous Transfer Mode (ATM) technology [11]. Broadband ISDN will enable high-speed delivery of digital continuous media over wide-area networks.

Conventional networks for continuous media are based on circuit-switched technology since circuit-switching allows for careful control of loss and delay. Integrated services networks, however, will use packet-based transport for digital voice and video services since packet-switching utilizes network resources more efficiently than circuit switching. The statistical multiplexing of packets exploits the fact that voice and video are inherently variable rate sources [7, 33]. The potential gain in efficiency is significant. Studies show 20-30% of the time in a voice conversation consists of silence [7], and the peak-to-mean ratio of variable-rate video is typically in the range of 1.5 to 4.5 [20].

Accommodation of continuous media streams in packet-switched networks represents a tremendous technical challenge. Distribution of continuous media streams is sensitive to network delays and variations in those delays, and these real-time requirements contrast with those for the applications that have traditionally dominated computer communications, e.g., bulk data transfer and remote login. In addition, most continuous media data has an inherent tolerance for limited data loss, unlike reliable data transfers. Current communication services in packet-switched architectures are ill-suited for distribution of continuous media, and new services to handle this traffic must be designed.

1.1 Quality in Network Distribution of Continuous Media

The distribution of continuous media across a packet-switched network requires consideration of all factors that significantly affect the quality of the playback at the receiving site. In this section we discuss issues important for maintaining high quality transmissions and the extant protocol methods for addressing these issues. The issues are encoding schemes, end-to-end network delays, network delay variations, and errors.

1.1.1 Encoding

In recent years considerable progress has been made in the design of efficient techniques for digital encoding of analog audiovisual data [28]. When distributed across a network, the quality of the reconstructed signal at the receiving site depends on the encoding scheme and the distortions introduced by network imperfections. In most applications this quality is ultimately judged by a human receiver and hence subjective metrics linked to human perception factors are used.

Performance criteria for encoding schemes include efficiency, sampling rate, complexity, and processing time [28]. The digital encoding of an analog signal involves sampling the signal with a certain period, known as the sampling rate. Efficiency is measured as the number of bits used per sample to represent the signal. Typical sampling rates for audio formats range from 8 kilohertz (kHz) for telephony to 48 kHz for compact disk quality sound. Common encodings for 8 khz voice, for instance, include the ubiquitous pulse code modulation (PCM) encoding at 8 bits/sample and adaptive differential pulse code modulation (ADPCM) encoding at 2-5 bits/sample.

Video sampling rates are often expressed in millions of pixels per second (Mp/s). One encoding for High Definition Television (HDTV) specifies resolution of 1280 pixels by 720 pixels with 60 frame/s to yield a sampling rate of 60 Mp/s [28]. The CCITT standard video format for visual telephony, the Common Intermediate Format (CIF) [38], has a spatio-temporal resolution of 360 pixels by 288 pixels by 30 frame/s that yields a sampling rate of 3 Mp/s The number of bits per pixel depends on many encoding-specific factors, e.g.,

the handling of color information. Uniform quantization at 8-24 bits/pixel yields enormous bandwidths and is therefore seldom used. In [28], state-of-the-art signal processing and compression techniques are estimated to generate on average much less than one bit/pixel/s for good quality video.

The sampling rate and efficiency of an encoding scheme determine the bandwidth required for network distribution. Since low-bit-rate encoding schemes result in a less precise reconstruction of the original analog signal, the selection of an encoding scheme represents a trade-off between consumption of bandwidth on the network and playback quality at the receiving site. For voice, a common technique for bandwidth reduction without a loss in quality is the suppression of transmissions during silence periods between speech activity periods. For video, sophisticated signaling processing techniques and compression of the encoded data are often necessary, though computationally demanding at the endsystems, due to the size of digital motion video streams.

Other dimensions of an encoding algorithm are the complexity of the algorithm and the processing time that it requires. The link between increased complexity and increased processing delay is platform-dependent. While software-based encoding for audio is typically feasible, at this time video requires hardware assistance for real-time encoding under most full-motion scenarios. Faster microprocessors and lower cost memory should enable software-based video encoding in desktop workstations within the next 5-10 years. In any case, whether an encoding algorithm introduces unacceptable delay is ultimately dependent on the delay requirements of the continuous media stream and what portion of the end-to-end delay can be allocated to the signal encoding/decoding process.

1.1.2 Roundtrip Delay

In an interactive continuous media session, human perception factors produce a requirement for bounded roundtrip delays. If roundtrip delays are too long, the interactive nature of the session is degraded. Quantifying this quality factor is difficult since individual human users may have different tolerances for delay and these tolerances will vary with the application.

Roundtrip Delay	Effect on Speech Quality
$>$ 600 ms	Conversation becomes difficult for untrained users.
400 – 600 ms	Conversation style increasingly affected by delays.
200 ms	Imperceptible if listener hears only from network, not off the air.
100 ms	Imperceptible even if listener is in the same room with speaker and hears off the air and from the network.

Table 1.1: Effect of Roundtrip Delays on Speech Quality.

The phenomenon has been studied extensively, however, for participants in interactive phone conversations [35], and these results provide first-order information on the delays acceptable in other interactive applications. Table 1.1 gives a summary of the effects of roundtrip delay on speech quality (adapted from [53]). In general, high-quality voice applications require less than 200 ms roundtrip delays, but delays of up to 600 ms have been shown to be acceptable [35]. Recent guidelines from CCITT suggest that even roundtrip delays of up to 800 ms have a limited impact on quality [10].

The requirement for bounded roundtrip delay is generally translated into a requirement for the network to bound the one-way end-to-end network delay. End-to-end delay guarantees are not generally available in current packet-based networks. Protocols for continuous media can offer some support for end-to-end delay constraints by limiting the amount of delay introduced by endsystem processing.

1.1.3 Delay Jitter

Statistical multiplexing of packets at internal network nodes introduces variations in the network delay experienced by individual packets. These variations are referred to as *delay jitter*. Delay jitter can lead to interruptions in the continuous playback at the receiver of the continuous media stream.

Current packet-switched networks do not provide jitter control in the network, though

there are proposals in the literature to do so [21, 61]. Jitter control in the network would relieve end-to-end protocols of delay jitter concerns, simplifying buffering and synchronization. Jitter control, however, implies per-packet processing at internal network nodes that estimates the delay bounds of packets and artificially delays any packets that are progressing too quickly through the network. Drawbacks to putting jitter control in the network thus include additional buffers at network nodes, management of timestamps on a per-packet basis, and nonwork-conserving scheduling disciplines that reduce network efficiency. In addition, jitter control in the network cannot address packet jitter due to endsystem effects such as variable network access delay in shared-media networks, operating system scheduling, and protocol processing.

Current protocols typically deal with delay jitter through buffering at the receiving site. When the first packet in a continuous media stream arrives at the receiver, the receiver does not begin playback immediately, but delays the playback for some time. Packets arriving from the network are held in a buffer, and this buffer provides protection from network delay variations. Resynchronization points in the continuous media stream allow the jitter buffer to be maintained.

A voice stream, for example, consists of an alternating series of speech activity periods, or *talkspurts*, and silence periods[6]. The natural resynchronization point in the stream is at each talkspurt boundary, and studies have shown that in general the adjustment of the playback of a talkspurt can be as much as 50% of the duration of the following silence period without affecting the quality of the playback [62]. The tolerance of packet voice for playback adjustment assumes that talkspurts are generally isolated from each other by relatively long silence periods, a property reported in empirical studies [6, 8]. To ensure this isolation, voice protocols enforce a minimum intertalkspurt time, or *hangover time*, when marking talkspurt boundaries [24].

1.1.4 Error Control

In a packet-switched network end-to-end protocols may need to protect the client from the misordering, duplication, corruption, and loss of packets in the network. Continuous media

streams require protection from misordering since the data stream carries time-ordered information. The end-to-end protocol can eliminate misordering and duplication through proper management of sequence numbers carried in packet headers.

Unlike the situation with reliable data transfers, the receiving site for a continuous media stream may prefer to receive packets with corrupted data, as opposed to having these packets discarded by the network. Encoding schemes in which each sample is encoded independent of adjacent samples, such as PCM-encoded voice, are robust in the face of bit errors in the data. End-to-end protocols can accommodate this requirement by providing an option to disable their data integrity checks, i.e., defeating their checksumming option.

Packet loss occurs due to bit errors and resource contention. The network transmission media is susceptible to random bit errors. In most networks, when a packet is corrupted in transmission, it is subsequently discarded by the data link layer protocol at the next receiving site. In fiber optic networks, random bit errors are rare, and packet loss is due to resource contention, that is, hardware buffers and switches can lose packets during periods of high load and transient periods of overload in the network. A fundamental trade-off for network designers is that the more aggressively the network uses statistical multiplexing of packets for bandwidth efficiency, the more likely packet loss will occur due to resource contention.

Most continuous media streams do not require reliable delivery, though their tolerance for packet loss is low. The impact of individual packet losses on the quality of a continuous media stream is variable since, in general, all bits in the encoded stream are not equally important. Techniques for robust signal processing in the presence of packet loss can significantly improve loss tolerances, but even the loss of a single packet may noticeably degrade playback quality at the receiver. The trend toward low-bit-rate encodings implies an increased importance on reliable delivery, though many extant protocols for continuous media [26, 43, 50, 59] do not provide any form of end-to-end error control.

It is important to recognize that error control in this context is fundamentally different from error control for data transfers. For traditional reliable data transfers, packet-level error control is handled in the end-to-end protocol by a handshaking protocol, typically an

Automatic Repeat Request (ARQ) scheme. Reliability is defined as in-sequence delivery of all the data injected into the network. For continuous media streams data completeness must be balanced with delay constraints. The purpose of error control mechanisms for these applications is to improve the quality of transmission from the viewpoint of the application by delivering more data in a timely fashion than delivery without error control could provide. Hence error control techniques must be evaluated in light of their impact on the overall quality of the continuous media transmission.

1.2 Quality of Service Networking and Error Control

Network services for continuous media streams must balance the trade-off between relying on performance guarantees, such as bounds on loss, delay, or throughput, if any, that can be provided by the underlying network and mechanisms to ensure transmission quality within end-to-end protocols. Quality of service (QOS) networks offer performance guarantees for individual channels. While deployment of first-generation QOS networks is imminent, the nature of supportable network performance guarantees is as yet uncertain. Many critical issues regarding quality of service networking are open research questions, and a variety of experimental techniques have been put forth, as surveyed in [36, 64].

Our work concerns the design of end-to-end error control. The demand for continuous media applications is developing much faster than the design and deployment of integrated service networks. There will thus continue to be considerable interest in the distribution of continuous media streams across conventional networking technologies. In our work we assume no performance guarantees from the network. If performance guarantees from the network are available, however, they can simplify the construction of end-to-end protocol mechanisms that we propose.

While it may be possible to engineer packet-switched networks to emulate the near-zero loss characteristic of circuit-switching [21], the loss of efficiency in the network will likely be too costly. It follows that end-to-end error control will always have a place in the network architecture, even if QOS networks become ubiquitous. Where QOS networks

are in use, reserving resources in internal network nodes is costly since these resources are shared by all the hosts attached to the network. End-to-end mechanisms that enhance the quality of service on a channel allow relaxation of the QOS needed from the network, thereby lowering cost.

An important class of next-generation networks are Asynchronous Transfer Mode networks, which transport data in small, fixed-size (53-byte) cells. Within the ATM community there are proposals for performance guarantees that provide bounds on cell loss over an ATM virtual circuit. Experience with the loss characteristics of ATM cell-based networking is not extensive, but mapping performance guarantees on cell loss bounds into loss bounds on higher layer data framing (packets) will be difficult. Specifically, multiplexing at the cell level implies that a loss of n ATM cells could cause the receiving endsystems to discard as many as n higher layer packets. Data in a higher layer packet is typically one or two orders of magnitude more than in the payload of an ATM cell, e.g., the current IP over ATM proposal has a maximum transmission unit (MTU) of 9000 bytes [14], while the payload of an ATM cell is at most 48 bytes. The implication is that very low loss rates at the ATM level can translate into relatively high observed loss rates for applications. This conflict is fundamental to the design of ATM networks. The fixed-size ATM cell is unlikely to be increased from its current size while the overhead of packet headers and the processing overhead at endsystems limits the feasibility of dramatically reducing the amount of data in each higher layer packet.

1.3 Inadequacies of Conventional Error Control

Conventional end-to-end protocols offer an all-or-nothing approach to error control. The first type of service makes no attempt to detect or recover packets lost in the network. The second type detects and recovers all packets lost in the network, i.e., 100% data completeness is assured. Neither service profile is well-suited to error control for real-time traffic flows.

In the Internet protocol suite, the User Datagram Protocol (UDP) offers the first type of service and the Transmission Control Protocol (TCP) offers 100% reliable delivery. As

researchers in the Internet community have begun to experiment with emerging applications such as continuous media, remarks such as the following conclusion reached at the 1991 Joint SIGGRAPH/SIGCOMM Workshop on Graphics and Networking [16] are representative:

> *TCP may not provide sufficient performance in some applications and does not support a rich enough set of quality of service.*
>
> *In particular, it would be desirable to have more control over the level of error detection and correction that is available from the transport layer[...] By allowing the application to select the amount of reliability supported by the transport layer, the TCP overhead incurred can be reduced to the minimum possible for that application, rather than the minimum possible for totally reliable transmission. The granularity available to applications at present is either TCP with relatively high overhead or UDP with relatively low functionality, and that is simply too coarse a grain for the wide spectrum of distributed applications which are now emerging on the Internet.*
>
> *One type of service that was mentioned by several graphics experts was* unreliable delivery with time constraints *for applications like real-time video and audio that can afford to lose some data but must have the next data delivered on time.*

One alternative to TCP is the connectionless transport, UDP. As a simple datagram service, UDP offers no segmentation/reassembly, no sequencing for stream data, and no flow control. Thus, like TCP, the UDP connectionless service is ill-suited to the needs of continuous media applications. From a practical standpoint, a common approach for designers of Internet-based tools to experiment with continuous media is summarized by the comments of the designers of the INRIA IVS videoconferencing tool [59]:

> *Neither TCP or (*sic*) UDP is ideal for IVS. UDP lacks reliability guarantees, while TCP has too many [...] delays generated by retransmission packets and multicast emission led us to use UDP and construct additional reliability services as necessary.*

This situation has led to a variety of proposals for new forms of end-to-end error control. In addition to protocols specific to the delivery of continuous media data [43], emerging general-purpose transport protocols such as XTP [57] and TP++ [5] are experimenting with novel error control mechanisms. In this literature (surveyed in Chapter 2), considerable emphasis has been placed on the open-loop techniques of adding redundancy to reduce the impact of losses (forward error correction) or of using priority channels in the network to protect the loss of important data (channel coding). Most researchers have dismissed altogether the viability of a retransmission-based approach. Our work challenges this viewpoint.

1.4 Thesis Statement and Overview

Our thesis is that retransmission of continuous media data often is, contrary to conventional wisdom, a feasible and effective strategy in most packet-switched networks. We define a new retransmission scheme that is architected to incorporate the delay sensitivities of the continuous media stream. Due to network transit delays, retransmission is not applicable under all network scenarios. Where applicable, error control based on delay-sensitive retransmission avoids the drawbacks of open-loop techniques, particularly the heavy demands on network resources imposed by forward error correction and the network-specific and/or encoding-specific factors underlying channel coding. When transmission quality requires it and open-loop error control techniques are available, however, retransmission can be used in conjunction with them.

The contribution of this thesis is a fundamental study of retransmission-based error control for delay-constrained stream traffic. Evaluation of delay-sensitive retransmission is largely done within the framework of a specific application domain—interactive packet voice. This approach was used for the following reasons. Packet voice is an important application domain with stringent performance requirements. In multimedia applications, the delay and error requirements for audio often dominate the requirements of other media streams. Also, interactive voice is an application for which the human perception delay

thresholds and the statistical characteristics of the bit stream generated are well studied. Finally, voice may be viewed as a simple first-order model for variable-rate video, whose traffic characteristics are not as well defined at this point in time. The components of the thesis may be summarized as follow.

- We define a novel retransmission-based error control technique for delay-constrained packet streams. We develop a simulation model and show that in a local area environment a significant amount of transparent error recovery through retransmissions is indeed feasible for realistic transmission scenarios.

- We develop an analytical end-to-end model for transmission of a delay-constrained packet stream. The model considers all of the protocol issues for maintaining the quality experienced by a continuous media stream and incorporates stochastic network delay behavior. We formulate a performance metric for continuous media retransmission and use the model to derive analytic expressions for this metric. These results allow the determination of retransmission effectiveness given the parameters of a transmission scenario, without lengthy simulations. This enables rapid evaluation of retransmission effectiveness under different assumptions on protocol and network parameters, which is particularly valuable for emerging network environments where traffic profiles and delay behavior are not yet well-understood.

- We provide empirical measurements of real packet voice transmissions across a large enterprise network. These empirical measurements are used to investigate to what extent the assumptions underlying our model reflect the complex dynamic behavior of a real network. The measurements also provide some insight into the applicability of retransmission-based error control in a network environment representative of current networking technology. Little empirical work appears in the literature on continuous media traffic over contemporary networks, and no previous study has specifically addressed packet voice over large, multiple-segment local area networks (LANs).

- Finally, Appendix A describes the definition, implementation, and evaluation of a

novel connection-oriented service that provides limited recovery from packet loss for delay-sensitive applications. The service is constructed as a modification to an existing service defined by a next-generation transport protocol, the Xpress Transfer Protocol (XTP). Our approach enables the application to control the aggressiveness of the underlying retransmission algorithm in order to provide a limited-retransmission service. A lightweight implementation is achieved through minor modifications to an existing implementation of XTP. Experiments using the service show error coverage through retransmission with significantly less delay than in a reliable connection. The relationship of this XTP-specific work to the retransmission scheme in the main body of the thesis is discussed in Section 2.3.1.

1.5 Document Organization

The remainder of the thesis is organized as follows. Chapter 2 surveys the literature on the mechanisms proposed to address the problem of error control for continuous media streams. Chapter 3 defines a novel retransmission scheme for packet voice and evaluates its feasibility through a simulation study. Chapter 4 presents our end-to-end transmission model for delay-constrained streams. We show derivations of analytic expressions for retransmission effectiveness across variations in transmission parameters. Chapter 5 reports on the results of an empirical study of actual packet voice transmissions across an existing network. We discuss the implications of these measurements for our end-to-end model and in applying our error control ideas to contemporary campus-wide LANs. In Chapter 6 we summarize our results and their impact on end-to-end protocol design for packet-based transport of continuous media streams. Appendix A provides the details of the design and implementation of a flexible, retransmission-based error control service that enhances an unreliable connection-oriented service proposed for the Xpress Transfer Protocol.

Chapter 2

Delay-Constrained Error Control

In this chapter we consider the range of extant error control techniques applicable to the distribution of continuous media streams over a packet-switched network. We first discuss the types of network imperfections from which continuous media clients need protection and the extent to which they are tolerant of these imperfections. Our work focuses on controlling packet loss, and we survey the techniques proposed for handling packet loss in a delay-constrained manner. At the end of the chapter we outline our retransmission-based approach to error control for continuous media and discuss its relationship to other proposed techniques.

2.1 Continuous Media Requirements for Error Protection

Packet-switched networks in general can corrupt, resequence, duplicate, and lose packets. We briefly discuss the requirements of continuous media applications for protection from these types of errors and the extant protocol mechanisms that can be incorporated into continuous media transports to deal with errors.

- Data corruption is rare but not unheard of in modern networks. For cell-switched networks, for example, data loss due to bit errors becomes negligible when the bit error rate is less than approximately 10^{-9}, and high-quality switches using fiber optic transmission exhibit bit error rates in the range of 10^{-12} to 10^{-15} [56]. In networks where the underlying transmission media is susceptible to impulse noise, e.g., atmospheric disturbances for radio waves, bit errors are a measurable source of packet corruption.

Continuous media streams are often tolerant of data corruption, and they may prefer that the network deliver, and not discard, corrupted packets. To provide this flexibility, in theory the end-to-end protocol need only have an option for disabling integrity checks over the user data. In practice, however, integrity checks at the data link layer are seldom optional, resulting in the discard of all packets corrupted in transmission. Since corruption above the data link layer occurs only in pathological cases, bit errors in contemporary networks are most commonly manifested as packet losses.

- Continuous media streams require that the network preserve sequencing since playback of the stream must be correctly sequenced in time. For end-to-end network services, sequencing is naturally handled at the packet level. While this is not strictly necessary, sequenced delivery is often preferred for some applications, e.g., in a video stream where out-of-order packet delivery within a frame buffer is acceptable within a frame buffer, but not across frame buffer boundaries. Protocols use sequence numbers to provide for sequenced delivery. Issues in properly managing the sequence number space include initialization of the sequence number space, handling sequence number wrap-around, and proper recovery from system failures to prevent aliasing. Sequence number management techniques, however, are well understood from experience with reliable transport layer protocols.

- Packet duplication in the network may occur due to retransmission of packets by endsystems using retransmission-based error control. Duplicates are identified and discarded by the receiver through the use of sequence numbers.

- Continuous media applications are generally able to tolerate some packet losses without a perception of degradation in service quality. The impact of packet loss depends on the amount of redundancy in the stream, the robustness of the decoding scheme, and the importance of the data lost. Digitized audiovisual streams exhibit a high degree of redundancy, a fact exploited by many encoding algorithms. Highly compressed or low bit-rate encoded streams have little redundancy and will be more sensitive to losses in the network. Decoding schemes that perform interpolation for missing or

corrupted data mitigate the effects of loss, though recursive encodings, such as differential PCM (DPCM) for voice, suffer substantial disruptions from a gap in the data since the predictive components at the decoder will lose synchronization: *error propagation*. Losses have a variable impact on quality for two reasons: measures of quality are subjective and all bits in the stream are not equally important in reconstructing the original signal. The quality of most continuous media streams is ultimately judged by a human user, and human perception factors thus influence the impact of particular losses. Information loss can adversely affect the decoding algorithm—e.g., error propagation—or the human user directly—e.g., the loss of an audio packet containing a key word such as "not". For these reasons even the loss of a single packet may be perceptible during playback.

The focus of this thesis is on controlling packet loss. Other types of errors are not considered further, and except where explicitly stated otherwise, from this point forward, the term *error control* in this document is synonymous with packet loss control.

2.2 Open-Loop Error Control

Much research has been focused on open-loop techniques that recover or limit the effects of packet losses during continuous media transfers without the roundtrip network delays associated with retransmission-based error control.

2.2.1 Forward Error Correction

Forward error correction (FEC) [33, 47] provides robustness in the presence of packet losses by adding redundant information to the original data stream. If only a small number of packets is lost, the added redundancy enables a reconstruction of the original data at the receiver.

There are two types of error correction codes [39]. A block code divides a message into coding blocks, containing both information and parities. These parities allow the receiver to reconstruct the information content of a block correctly, provided enough of the block gets

through uncorrupted. A convolutional code operates under a continuous bit stream model. At the transmitter, the constraint length determines how many bits of past information are used to determine the parities; at the receiver the constraint length determines when to decide about the decoded information. The ratio of parity bits to total bits is known as the degree of *overcoding* for the code.

Forward error correction is often identified with transmission-media level error handling. In the language of coding theory [3], an error is defined as a corrupted bit (or symbol) with an unknown value in an unknown location. An erasure is a corrupted bit (or symbol) with an unknown value in a known location. Most extant error correction schemes were designed to detect and/or correct random errors in a bit stream that exist in unknown locations, e.g. for noisy media such as satellite or wireless links [9, 32, 37]. Fiber-optic packet-switched computer networks require a different error model. While random errors are rare, losses occur on packet boundaries (burst erasures), and thus burst erasure correcting codes are the appropriate form of forward error correction for these environments.

The effectiveness of burst erasure codes is strongly dependent on the burstiness of the network loss process and the degree of overcoding. In [52] a block code for cell-switched networks based on the exclusive-OR operation was used to generate one or two redundant cells in a block of k cells. The study concluded that the coverage from this code was not very useful and suggested the need for sophistication in multiplexing and/or cell dropping at the cell switches to improve the performance of the FEC code. In [47] a two-dimensional cell-level exclusive-OR scheme is applied to traffic traveling on the same virtual path in an ATM network. The scheme is evaluated using a simple two-state Markov model for the cell discard process at a switching node. Dramatic reductions in cell loss rates are reported, though the degree of overcoding used requires an estimation of the likelihood of consecutive cells from the same virtual path being lost during congestion. The probability of multiple cell losses within the same block is assumed to be low—the probability of consecutive cell losses is varied between 0.01 and 0.1 for low-bandwidth virtual paths, and between 0.1 and 0.4 for high-bandwidth virtual paths.

A primary consideration with FEC schemes is the feasibility and cost of hardware

implementations since even moderate bandwidths require hardware support. The modified Reed-Solomon code presented in [39] was designed for VLSI implementation at speeds up to 1 Gbit/s, and the hardware complexity is independent of the block size. The code takes k data symbols as input and produces $k + h$ data symbols as output. For a single chip implementation operating at 400 Mbits/s the code is limited to $h = 16$ with 8 bits/symbol and for operation at a data rate of 1 Gbit/s $h = 4$ with 32 bits/symbol. The parameter k is variable.

Packet loss in the network results from congestion. The increase in bandwidth due to FEC overcoding adds to network traffic and therefore increases congestion. Thus, the use of forward error correction causes additional packet losses in the network. In [4], this trade-off between loss recovery and additional loss is studied using the Reed-Solomon burst erasure code just described. The network model is an ATM multiplexer with a set of 32 statistical sources. Sources are of two types, *bulk sources* representing bulk data transfers, or *video sources* representing video streams. Bulk sources are characterized with a statistical model while the traffic from video sources is based on an actual trace of a variable-rate encoded movie. Three scenarios are presented. In the scenarios studied for homogeneous sources, i.e., all bulk sources or all video sources, the FEC code was only marginally effective. With a mixed scenario of 8 bulk sources intermixed with 24 video sources, the FEC algorithm was judged advantageous, with a suggested overcoding of 14%. One conclusion of the study is that the number of sources in a network using FEC may have to be limited in order for the effect to be significant.

2.2.2 Hybrid FEC/ARQ

Many hybrid schemes in which forward error correction is combined with the use of retransmissions have been studied [12, 32, 49], though only in the context of channel efficiency, and not for delay-constrained communication. Under these schemes the receiver uses the parity bits in forward error correction to recover from losses, but when there are too many errors a retransmission is requested from the transmitter. The retransmission may be the original message [32] or additional parity bits that will allow recovery of the original message [12].

Hybrid schemes can be effective in reducing the bandwidth consumed through over-coding since larger losses can be recovered through retransmissions. Overall performance of the error recovery mechanisms as measured in terms of average throughput is generally superior to pure FEC or pure ARQ [32, 49]. The drawbacks of hybrid FEC/ARQ are the requirement for FEC hardware and the complexity of coordinating the retransmission scheme with the FEC hardware. This coordination may not be cumbersome for data link protocols employing a FEC/ARQ scheme, but protocols handling the transport of continuous media streams are generally not at the data link. These protocols must then have an interface that allows them adequate control over the FEC hardware in order to take advantage of proposed hybrid FEC/ARQ schemes.

2.2.3 Channel Coding

Channel coding refers to a class of approaches that use multiple network channels to transmit the components of a single encoded stream. A simple example of channel coding is *odd-even* bit interleaving [29] for a voice transmission. For each B voice samples the odd-numbered bits are stuffed into one network channel, and the even-numbered bits into another channel. Unless both the odd and the even packets are lost, missing odd (or even) samples can be recovered with simple nearest neighbor interpolation using the surviving odd (or even) samples.

The advantage of the technique is that, if the two network channels have a statistically independent packet loss probability of ρ, the probability of losing both the odd and the even packet of a sample is ρ^2. Drawbacks include additional processing overhead and a synchronization delay of B samples at the decoder. Also, nearest neighbor interpolation is only effective with non-recursive encodings such as PCM since residual signals must have a strong auto-correlation.

Channel coding techniques generally exploit the hierarchy of data importance created by embedded or layered encoding, which are techniques that produce successive approximations to a signal. Examples include sub-band, pyramid, and transform coding [30, 33]. Signal components that contribute relatively little are sent with low priority in the network

while important signal components are given preference in the network. Specifically, in networks such as ATM networks, the channel carrying the most significant components can be given higher priority in the network when congestion forces packet discard.

In the context of ATM networks, a number of techniques have been proposed [20, 33, 55, 63] to split signals into separate channels and use priority channels in controlling quality degradation due to network loss. For example, one study on voice [58] has investigated a least significant bit LSB dropping scheme. Each set of B samples is divided into two segments with the six significant bits of each sample in one segment and the two least significant bits of each sample in the other. Using subjective scoring from a set of human listeners, acceptable quality is reported at 2-5% loss on the LSB-channel for 32 kbit/s embedded ADPCM voice and at over 5% for PCM-encoded voice. These results assume a packet size of 32 bytes, or 4 ms for PCM and 8 ms for ADPCM.

In [20] a detailed investigation of a two-priority network model is presented using simulations with actual Discrete Cosine Transform (DCT) encoded voice and video as input. Under the model, the quality of the stream is shown to be quite sensitive to the fraction of traffic allocated to the high priority channel, and a method to adapt the channel distribution based on feedback on network conditions is developed. A drawback to this study and to channel coding in general is the assumed sophistication of the underlying network in supporting selective dropping of packets during periods of congestion. This functionality is not available in current LANs, and its implementation cost in high-speed cell switches is of unknown complexity at this time. The methods of [20] assume a time-out driven loss mechanism at the internal network queues, which requires sequential checking or sorting of the packets by a time-to-live field. It is suggested that as VLSI technology improves it will become feasible to use parallel hardware to actively discard packets that have timed out in the queue.

2.3 Retransmission-Based Error Control

Most researchers have dismissed retransmission altogether as a form of error control for continuous media. Two attempts to adapt retransmission to the needs of loss-tolerant, delay-sensitive traffic are presented in this section. These approaches construct connection-oriented communication services parametrized by application-specific error tolerances. Due to latency considerations, all packet losses in the network are not recovered, and the user of these services is expected to tolerate occasional delivery of partially corrupt buffers from the network service. In both cases the underlying ARQ schemes avoid timer-based detection of packet losses in the network and go-back-n retransmissions. By contrast, both of these mechanisms are part of the ARQ algorithm for the conventional transport protocol TCP.

2.3.1 Partially Error-Controlled Connections

A next-generation transport protocol, the Xpress Transfer Protocol [57], provides a novel unreliable stream-based service, referred to as *no-error mode*. The service provides sequenced delivery with the full functionality of an XTP association, e.g., rate control, an out-of-band data channel, and multicast, but with error control disabled. In [15] a new XTP service for Partially Error-Controlled Connections (PECC) is proposed that generalizes the *no-error mode* service to allow for application-specific parameterization of the underlying XTP error control algorithm.

The PECC service enables limited recovery of packet loss for stream-based communications in which data completeness must be traded off for low delay service. The application provides a description of the frequency and density of losses that it expects to be able to tolerate, and the underlying communication service uses this information to parametrize its retransmission algorithm. Parameter settings exist for the emulation of the limiting services, i.e., either a *no-error mode* connection or a 100% reliable connection.

The PECC service is properly viewed as an enhancement to the XTP *no-error mode*, which has no active error control, since the PECC service provides no guarantees to the user on the amount of corrupted data the user will be delivered. If the application-specified

error bounds are exceeded, an indication of this violation is returned to the user but the data is not recovered. The underlying assumption is that a temporary violation of the error constraints of the stream is more desirable than excessive delays to recover the lost data. The PECC service interface, however, gives an application the capability to increase or tighten its error tolerance on an active network connection, if network loss becomes too great.

A design goal underlying the PECC service was to demonstrate that application-specific error control could be constructed by small modifications to an existing protocol. By exploiting functionality implemented for reliable modes of the protocol, particularly management of retransmission buffers, the PECC service was constructed in a very lightweight manner. Its implementation required only approximately 100 lines of additional code in an existing XTP implementation, and the code modifications were only at the XTP receiver. A more complete description of PECC, its implementation, and experimental results are given in Appendix A of this thesis.

2.3.2 Application-Oriented Error Control

The application-oriented error control (AOEC) presented in [23] has the objective of satisfying an application's error tolerance with minimum retransmission overhead. Like the PECC service, an application using the AOEC service interface gives a description of its maximum error tolerance that parametrizes retransmission decisions in the protocol. Unlike the PECC service, however, the AOEC scheme guarantees that the maximum error tolerance of the application is always respected by retransmitting lost data whenever necessary.

A key difference between PECC and AOEC is that the latter has some knowledge of the structure of the application data stream. Specifically, the AOEC service interface is based on the notion of a *segment*, defined as the smallest unit of data that the application accesses independently. Applications specify their requirements in terms of the maximum number of losses per segment and the maximum size of any burst loss within a segment.

The underlying retransmission algorithms for the AOEC service are more complex

than the XTP algorithms on which PECC relies. Both positive and negative acknowledgments appear in the AOEC protocol and three sequence numbers are carried in each packet. The sequence numbers identify the segment number, the offset within the segment of this packet, and a *shipment* number to detect losses during retransmissions. The AOEC receiver sends requests for retransmissions to the transmitter based on the total amount of data received for a segment and the user-specified tolerance for loss within a single segment. Data stream events, e.g., a packet arriving out-of-order, trigger retransmission requests. For robustness the receiver uses an additional timer to detect very long burst losses during which no subsequent packets will arrive to enable loss detection.

The analysis in [22] characterizes the AOEC protocol in terms of average end-to-end delay of a segment, maximum end-to-end delay of a segment, and average throughput for a stream. Through analysis and discrete-event simulation, expressions for these measures are obtained, and the performance of AOEC is shown to compare favorably with the throughput of SNR [46], an experimental transport with timer-driven control algorithms. The performance analysis does not model stochastic network delays.

2.4 Delay-Constrained ARQ

This thesis focuses on the definition and evaluation of a novel retransmission-based error control scheme for continuous media streams. This scheme is parametrized by the delay sensitivities of the continuous media stream, and not by a statically defined application tolerance for gaps in the data, as with the retransmission-based error control services discussed in Section 2.3. Our analysis of retransmission effectiveness uses a performance metric that measures the quality of the transmission for the continuous media stream. It considers the performance of the retransmission scheme in light of the interaction between stochastic network delay behavior and timely playback of the continuous media stream at the receiving site.

Possible drawbacks to a retransmission-based approach are the latency penalty for recovering packet losses and buffering requirements at the transmitter in networks with high

bandwidth-delay products. Retransmission is indeed not applicable for delay-constrained traffic in networks with very long one-way delays, e.g. geosynchronous satellite links. We note, however, that propagation delay for a high-quality fiber-optic link from New York to San Francisco is approximately 15 ms in one direction. We contend that this amount of delay does not *a priori* eliminate the possibility of using a retransmission-based approach to error control. With high-speed switching and powerful endsystems, it is reasonable to assume that packet loss can be discovered and retransmitted within a 50-150 ms time frame, which is within the tolerance of many interactive continuous media applications. More importantly, this scenario represents an extreme—the majority of applications will span much smaller geographic areas. As for buffering considerations, they are not a concern for low-bandwidth traffic, which includes virtually all audio formats and some low-bandwidth video formats. For video streams with bandwidths of a few Mbits/s, the retransmission buffers maintained at the transmitter are on the order of tens of kilobytes. The system cost of such buffers will continue to decrease with the price of memory chips. We argue that, while not feasible in all cases, there are a significant number of transmission scenarios for video streams in current systems under which retransmission buffering represents a reasonable system cost.

In order to focus on pure retransmission effects, we do not consider the effects of combining retransmission with channel coding or forward error correction. We attempt to quantify only the error coverage provided by delay-sensitive retransmissions. When retransmission alone cannot meet the quality requirements of the application and open-loop methods are available, we expect hybrid techniques to be used, although a pure retransmission-based approach is attractive in its portability and low overhead, as compared with open-loop techniques.

Chapter 3

A Delay-Constrained Retransmission Scheme for Packet Voice

We introduce a novel approach to error recovery for continuous media protocols and specifically for packetized voice protocols. This scheme is referred to as *Slack Automatic Repeat Request* or *Slack ARQ*, because the technique involves managing the buffer time, or slack, of packets at the packet voice receiver. In this chapter we develop a simulation model for packet voice transmission over a loaded Fiber Distributed Data Interface (FDDI) local area network and use it to establish the feasibility of Slack ARQ for packet voice in this environment.

The remainder of this chapter is structured as follows. In Section 3.1 we present our Slack ARQ scheme. In Section 3.2 we develop a simulation model of our error recovery scheme for a local area network environment. Our simulation takes into account the delays incurred at the endsystems due to operating systems scheduling and protocol processing. We present several simulation experiments to evaluate the effectiveness of Slack ARQ. In Section 3.3 we state our conclusions from this study.

3.1 Slack ARQ

Experiments with the transmission of packetized digital speech across computer networks date to the earliest days of packet-switching, with a network voice protocol (NVP) for the

Internet having been specified in 1976 [13]. The feasibility of high-quality voice transmission for large numbers of users has dramatically risen in the past five years with the advent of high-speed LANs and powerful desktop computers with audio hardware as a standard feature. Protocols and conferencing tools have become available for multiple-party connections. Examples of software from the research community include NVP, the *vat* protocol [26], and the NEVOT audioconferencing tool [50].

In a local area network, delay and delay jitter is introduced by variable network access delay, operating system scheduling, and protocol processing at the endsystems as well as congested store-and-forward routers if the network has multiple segments. Packet losses due to transmission errors are rare in a local area network, but hardware buffers and routers can lose packets during periods of high load and transient periods of overload in the network. Extant protocols have mechanisms to address packet delays that threaten quality, but packet losses are not recovered.

The principle behind the Slack ARQ technique is to extend the buffering strategy commonly used in packet voice protocols to handle delay jitter such that timely retransmissions of lost packets becomes feasible. We show that this extension can provide a high probability of successful error recovery using retransmissions for realistic packet voice scenarios in a local area network setting. In this section we first describe buffering techniques for delay jitter reduction and then the Slack ARQ technique.

3.1.1 Delay Jitter Reduction

If the network delay of voice packets is not constant, i.e., packets are subject to delay jitter, the receiver may observe *gaps*, which result in interruptions in the continuous playback of the voice stream. Delay jitter in packetized voice transmission is commonly addressed through a *control time* at the receiving system. The first packet in a voice stream is artificially delayed at the receiver for the period of the *control time* in order to buffer sufficient packets to provide for continuous playback in the presence of jitter. Note, however, that the the control time cannot be arbitrarily large due to constraints on the end-to-end delay. Since voice data consists of an alternating series of *talkspurts* and *silence periods* and since talkspurts are

generally isolated from each other by relatively long silence periods [6, 8], voice protocols typically impose the control time on the first packet of each talkspurt.

We refer to the *playback time* of a packet as the point in time at which playback of the packet must begin at the receiver in order to achieve a zero-gap playback schedule for the talkspurt. We refer to the *slack time* of a packet to denote the time difference between its arrival time at the receiver and its playback time. Note that, independent of whether a control time is used or not, the arrival of the first packet in a talkspurt at the receiving site determines the start of the zero-gap schedule and thus the playback time of all packets in the talkspurt. Due to delay jitter, a packet may arrive before or after its playback time. In the former case, the packet is placed in a queue, the so-called *packet voice receiver* (PVR) queue, until it is due for playback. In the latter case, a gap has occurred and the packet is played back immediately. [1]

In Figure 3.1 we illustrate the occurrence of gaps due to delay jitter and the elimination of gaps through the introduction of a control time. The time sequence shown in Figure 3.1 illustrates the transmission of a talkspurt consisting of 5 packets. The four timelines pictured represent, from top to bottom, packet generation at the voice source, packet transmissions at the sender, packet arrivals at the receiver, and the playback of voice samples at the receiver. Packet generation is taken to occur periodically, at the end of a fixed-size packetization interval. Packets are indicated by a vertical bar. Note that in a high-speed network the transmission time of a packet is negligible compared to the length of a packetization interval. In Figure 3.1 the delay incurred by a packet due to protocol processing, scheduling delay, and media access delay is indicated by the arrows from the second timeline (Packetization) to the third timeline (Arrival at the Receiver). We assume that propagation delay in a LAN is negligible. At the bottom of Figure 3.1 we present two scenarios for the playback of the voice packets. Scenario (a) depicts playback without a control time, scenario (b) with a control time. In scenario (a) two gaps are observed, one between the first and second packet, and another between the third and fourth packet. In scenario (b) the presence of a

[1] An alternative policy is to play back only that part of a late packet that has arrived within its playback time, thus ensuring that the talkspurt has the same duration at the receiver as it did at the source.[18]

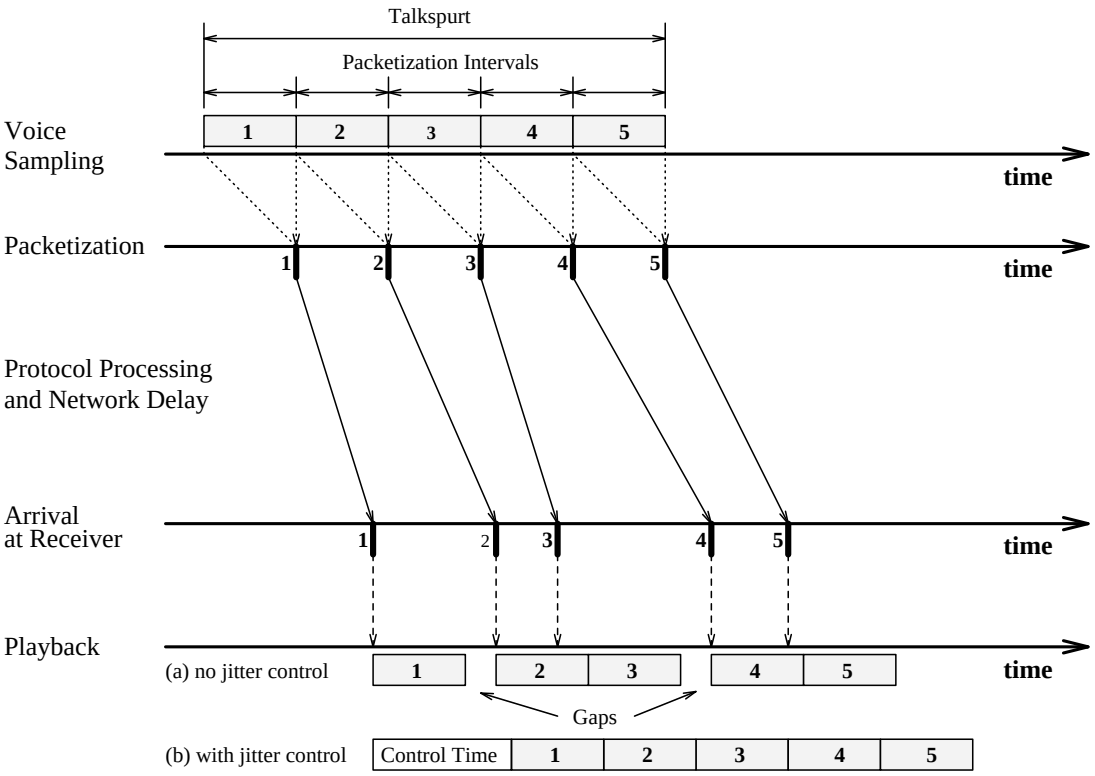

Figure 3.1: Transmission and Playback of a Talkspurt.

control time delays the first packet of the talkspurt and eliminates gaps. Thus, in scenario (b), the playback of the voice samples is continuous.

A packet voice protocol with control times must be able to identify the beginning of talkspurts. This can be done explicitly with a dedicated bit or with a combination of timestamps and sequence numbers. Determination of the control time is more difficult since it requires knowledge of the network delay distribution. Note that to eliminate gaps completely the control time must be set equal to the maximum variation of the network delay. Numerous methods have been proposed for estimating the control time of a talkspurt, based on network delay measurements [42], on stochastic assumptions of the network delay [1, 2], or both [45]. One result of note for our work is that suitable control times were found to be in the range of 2-3 times the mean network delay in one study [45].

3.1.2 Extending the Control Time for Timely Retransmission

Slack ARQ is an error control scheme based on the retransmission of lost packets. Recall that due to the need for continuous playback of voice packets, retransmitted packets must arrive before they are due for playback. The principle behind Slack ARQ is to extend the control time at the beginning of a talkspurt and use the extended control time at the PVR such that the slack time (as defined in Section 3.1.1) of arriving packets is lengthened. With this simple mechanism timely retransmissions of lost packets can be achieved with a high probability.

In Slack ARQ, whenever a lost packet is detected, the receiver requests a retransmission of the missing packet. The packet voice receiver assumes that a packet is lost if it receives a packet out of sequence. Note that the packet voice receiver will wrongly assume that a packet is lost if packets are misordered in the network. However, the misordering of packets very rarely occurs in broadcast LANs. If the retransmission is attempted but the packet is lost or late, the packet voice receiver does not hold back subsequent correctly received packets nor does it attempt any additional retransmissions of the lost data. Therefore, Slack ARQ does not guarantee that lost packets are successfully recovered. The percentage of retransmissions that are completed successfully is largely dependent on the appropriate choice of the control time. Note that the likelihood of successful retransmissions decreases if consecutive packets are lost.

We illustrate the advantages of Slack ARQ in Figure 3.2. The figure depicts the same transmission scenario as in Figure 3.1, i.e., the transmission of a talkspurt consisting of five packets. Here, however, we assume that the second packet of the talkspurt is lost. At the bottom of Figure 3.2 we show three scenarios. For all scenarios we assume the existence of jitter control with an appropriately selected control time. Scenario (a) shows error handling typically found in extant voice protocols, i.e., no retransmissions are attempted. If a packet is lost, playback of the lost packet is skipped. As shown in Figure 3.2 skipping the playback of a lost packet results in a gap equal to the time of the packetization interval. In scenarios (b) and (c), the receiver requests a retransmission of the lost packet. In both scenarios

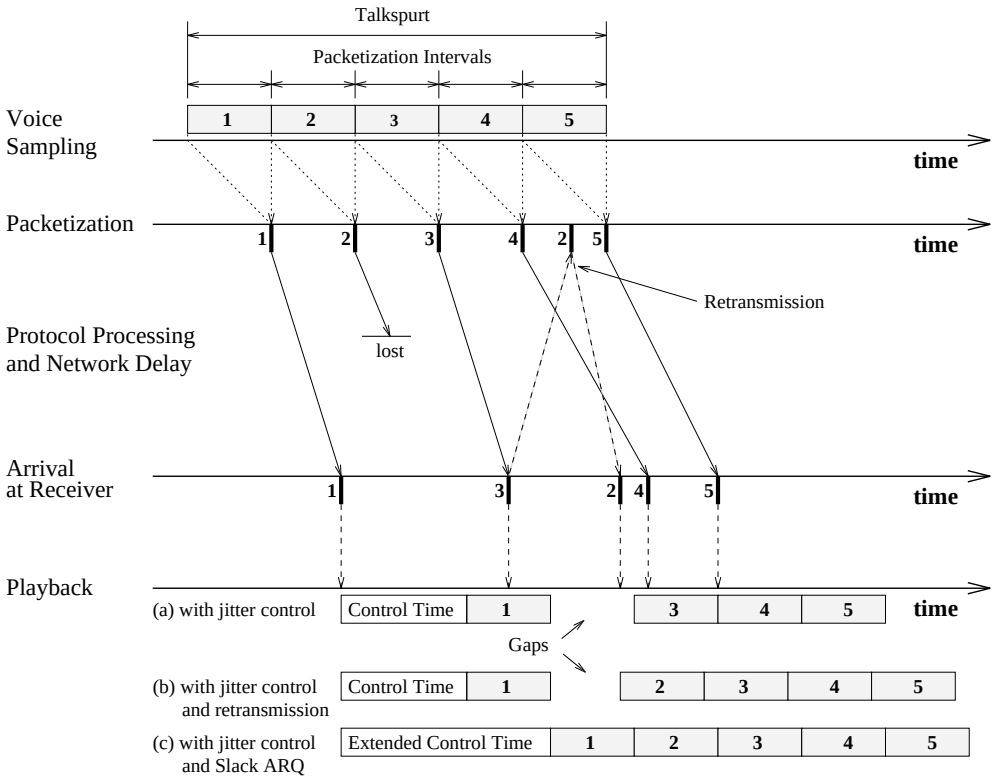

Figure 3.2: Transmission of a Talkspurt in the Presence of Errors.

the lost packet is detected upon arrival of the third packet to the receiver. Scenario (b) shows the drawback of a retransmission scheme without Slack ARQ. If a packet is lost, playback of all packets is discontinued until the retransmission has been completed. Since the retransmission of the second packet is not completed before its playback time, a gap is observed. In scenario (c) we assume a Slack ARQ scheme. Due to the extended control time, the retransmission of the second packet is completed before its playback time.

Slack ARQ depends on appropriately extending the control time of the PVR queue in order to increase the slack time of arriving packets. Packet voice protocols provide control times to account for delay jitter, but these control times do not consider the roundtrip time of the network, which is central to packet retransmissions. In calculating control times for jitter purposes, packet voice protocols must estimate the delay variation of packets traversing the network, either through performing measurement of this variation or through

the use of network-specific constants. The additional amount of control time required for Slack ARQ can be calculated using these same mechanisms to derive estimates for the network roundtrip delay. Hence, the implementation of slack ARQ in an extant or future packet voice protocol will add little overhead. Note that in the case where no packets in the talkspurt are lost, the extended control time for Slack ARQ provides an additional measure of protection from gaps in the playback due to delay jitter.

Note that Slack ARQ can be improved in the sense that retransmissions are suppressed when the probability of timely retransmission is low. In this case, before Slack ARQ requests retransmission of a lost packet it would examine the length of the PVR queue and compare it against the measured or estimated retransmission time.

A possible drawback of Slack ARQ is that it increases the end-to-end delay of all voice packets. Recall that the end-to-end delay in interactive voice transmissions significantly affects the speech quality (see Table 1.1 in Chapter 1). Therefore, before any refinement to the Slack ARQ schemes are given, e.g., for calculating the extended control time and thresholds for retransmission suppression, one needs to show that the increase in end-to-end delay due to Slack ARQ does not interfere with requirements for speech quality. In the next section we will show that in a LAN environment, Slack ARQ can successfully recover from almost all single packet losses in a talkspurt with only modest extensions to the control time used for jitter control.

3.2 Evaluation of Slack ARQ

In order to evaluate the effectiveness of our Slack ARQ scheme we present a simulation model of digital speech distribution across a local area network. We conduct experiments with the simulation model and provide answers to the following questions:

- *How much control time is needed for Slack ARQ to ensure a high probability of successful retransmissions?*

  Note that the control time at the PVR results in increased end-to-end delays for all packets. However, since voice transmission is sensitive to end-to-end delay, the control

time cannot be increased arbitrarily.

- *How does the control time required for jitter control compare to the control time required for Slack ARQ ?*
  We expect that our Slack ARQ scheme will not be acceptable for extant packet voice protocols if the control time needed for Slack ARQ is not in the same order of magnitude as the control time used for jitter control.

- *How does Slack ARQ perform if consecutive packets are lost?*
  Due to the simplicity of the error detection scheme in Slack ARQ we expect the effectiveness of Slack ARQ to degrade if consecutive packets are lost. However, Slack ARQ should be able to recover at least partially from consecutive packet losses.

- *How sensitive is Slack ARQ to particular network delay distributions ?*
  Although we expect the effectiveness of Slack ARQ to vary for different network delay distributions, Slack ARQ should be applicable for a wide range of network delay distributions.

In Section 3.2.1 we discuss the parameters of our simulation model. Then we present a set of experiments to evaluate the Slack ARQ scheme for different sets of system parameters.

3.2.1 Simulation Model

For our simulation model we chose a local network environment with multi-user workstations as endsystems and an FDDI network as the communication network. Since we are primarily interested in studying the behavior of the queue at the packet voice receiver queue, we model a single uni-directional voice channel. The voice traffic stream is modeled as alternating talkspurts and silence periods whose lengths are exponentially distributed with means 350 ms and 650 ms, respectively. These figures represent a talk activity of 35 percent as suggested in [8]. The packetization interval, $T_{\overline{x}}$, determines the duration of speech (in milliseconds) captured by each network packet. The number of packets in a talkspurt will be the length of the talkspurt in milliseconds divided by $T_{\overline{x}}$.

The one-way delay of the network, T_{net}, consists of three components: protocol processing at the sender, T_{sndr}, network access delay, T_{access}, and protocol processing at the receiver, T_{rcvr} :

$$T_{net} = T_{sndr} + T_{access} + T_{rcvr} \tag{3.1}$$

For simplicity, the processing characteristics of sender and receiver are assumed to be equivalent. The processing delays T_{sndr} and T_{rcvr} are assumed to consist of a fixed component C_{const} and a variable component C_{var} as given here:

$$T_{sndr} = T_{rcvr} = C_{const} + C_{var} \tag{3.2}$$

The fixed component C_{const} represents the minimum time required for processing a packet at the endsystems. For representing the variable component of the processing delay we choose a truncated Cox distribution $\Phi(x_1, x_2, x_3)$ with mean value x_1, coefficient of variation x_2, and a maximum value of x_3. Selecting values from a Cox distribution for the variable delay allows us to consider different degrees of variances of the distribution. For example, selecting $x_2 = 1$ as the coefficient of variation will yield an exponential distribution, values $x_2 < 1$ will result in an Erlang distribution with low variations, and $x_2 > 1$ will result in a highly variable hyperexponential distribution. Since system measurements show that values for processing delays do not grow arbitrarily large, we select a truncated distribution. The minimum processing delays of packets at endsystems is assumed to be low, i.e., we set $C_{const} = 1$. However, due to multitasking at the endsystem the packet processing delay is highly variable. Therefore, we select the variable component of processing delays T_{sndr} and T_{rcvr} from a truncated Cox distribution, specifically $\Phi(1, 6, 10)$.

We assume that the FDDI network is heavily loaded. Network measurements show that non-negligible packet error rates in FDDI networks are observed only under conditions of high network load. We assume that all voice data is transmitted as synchronous traffic. With the recommended default value for the Target Token Rotation Time (TTRT) of 8 ms [27] an upper bound of the access delay is given by $2 \cdot TTRT$ [31]. Assuming that T_{access} is exponentially distributed, we obtain a truncated Cox distribution $\Phi(8, 1, 16)$ for the network access delay.

Parameter	Description	Min.	Max.	Avg.
T_{net}	One-way network delay	2	36	12
T_{sndr}	Processing delay at sender	1	10	2
T_{rcvr}	Processing delay at receiver	1	10	2
T_{access}	Network access delay	0	16	8
T_V	Control time	-	150	-
$T_{\overline{x}}$	Packetization interval	-	150	-

Table 3.1: Default Parameters of the Simulation Model (in milliseconds).

The key parameters of a packet voice protocol are the control time at the PVR queue, T_V, and the packetization interval, $T_{\overline{x}}$. Both T_V and $T_{\overline{x}}$ are constrained by the end-to-end delay requirements of the channel. To achieve a voice channel with high speech quality, we do not consider values of $T_V > 150$ ms and $T_{\overline{x}} > 150$ ms.

The error model of our simulation model arbitrarily generates a period of time in the talkspurt where packets are dropped, the so-called *error period*. The start of the error period is uniformly distributed among the duration of the talkspurt. There is at most one error period in a talkspurt. However, in one error period multiple consecutive packets may be lost. An error period in which one packet in a talkspurt is lost is called a *1-error*, an error period in which two consecutive packet are lost is called a *2-error*, and so on. In the case of a loss of multiple consecutive packets, the receiver sends a single request for the retransmission of lost packets. This is a realistic assumption since FDDI frames are large relative to the amount of data in a voice packet. In Table 3.1 we summarize the default parameters of the simulation model.

3.2.2 Experiments

The simulation model was developed using the SES/Workbench simulation package [51]. All simulation experiments were run on a Sun Sparc2 workstation. Each data point represents

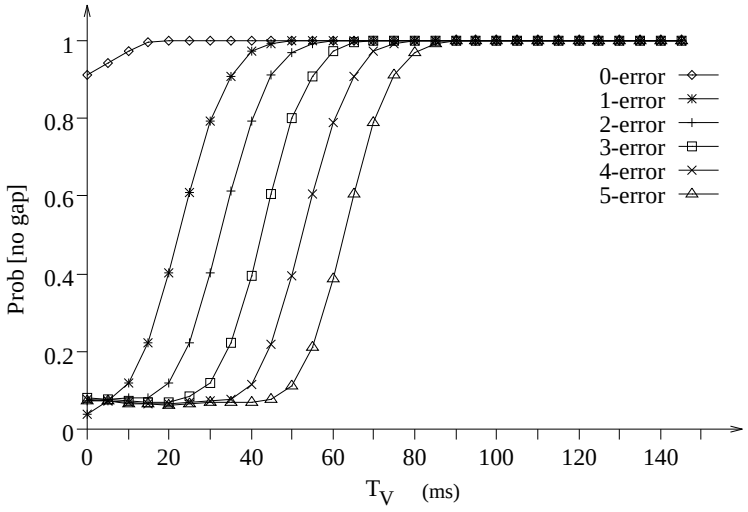

Figure 3.3: Probability of Continuous Playback ($T_{\overline{x}} = 10$ ms).

the observation of at least 6000 talkspurts.

We present three sets of experiments. In the first two experiments, we investigate the degree to which packetization interval and control time affect the ability of Slack ARQ to recover lost packets. In the third experiment we examine the sensitivity of our Slack ARQ scheme towards variations in the network delay behavior. Our performance measure is the probability $Prob[no\ gap]$, the probability that no gaps are observed during the playback of a talkspurt. If the talkspurt contains packet losses then $Prob[no\ gap]$ is equivalent to the probability of successful retransmissions. We show the values of $Prob[no\ gap]$ for single and multiple packet losses. For reference, we always include the simulation results of $Prob[no\ gap]$ for error-free voice transmissions.

3.2.2.1 Experiment 1—Control Time

Figures 3.3, 3.4 and 3.5 show the effect of increasing the control time both on jitter control and on the probability of successful retransmissions using the Slack ARQ scheme. The packetization interval is assumed to be constant and is given by $T_{\overline{x}} = 10$ ms in Figure 3.3, $T_{\overline{x}} = 25$ ms in Figure 3.4, and $T_{\overline{x}} = 50$ ms in Figure 3.5.

In each figure, the curve labeled *0-error* depicts the probability that in an error-free

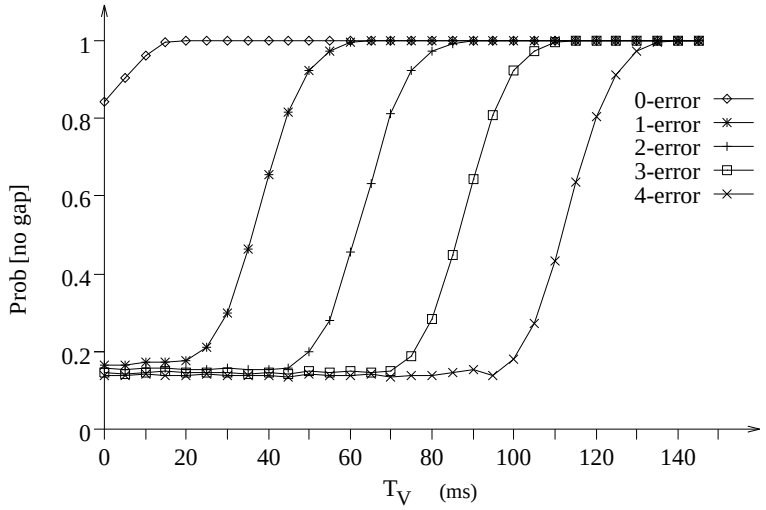

Figure 3.4: Probability of Continuous Playback ($T_{\overline{x}} = 25$ ms).

transmission a packet arrives at the receiver before its playback time. The other curves show the probability that Slack ARQ will successfully recover from the loss of n consecutive packets, where $1 \leq n \leq 5$. Thus, the curve labeled *1-error* in Figure 3.3 represents the probability of recovery from single packet losses.

Note that without a control time, i.e., $T_V = 0$, the probability that no gaps occur ranges between $75 - 91$ percent for all simulation runs in this experiment. A control time of $T_V = 15$ ms is needed to reduce the probability of a gap in the playback schedule to less than 1 percent. Short control times of $T_V \leq 15$ ms result in a low probability of error recovery, only 20 to 30 percent for the *1-error* case.

The extended control time necessary for Slack ARQ to be effective can be directly obtained from Figures 3.3, 3.4, and 3.5. In Figure 3.4, for example, complete coverage of single error losses is achieved with a control time of $T_V \geq 55$ ms. The following argument explains the observed behavior of Slack ARQ. Define the *virtual slack* of a lost packet as the slack time of that packet if it had not been lost, but had instead arrived at the receiver. When a single packet is lost, the slack time at the receiver of the packet following the lost one is approximately the virtual slack of the lost packet minus one packetization interval, $T_{\overline{x}}$. In order for a retransmission to occur before the playback time of the lost packet, the slack

time of the out-of-sequence packet must be greater than a roundtrip time in the network, i.e., $2T_{net}$. Thus, the virtual slack of the lost packet should have a value of $T_{\overline{x}} + 2T_{net}$. For example, in the simulations shown in Figure 3.4, $T_{\overline{x}} = 25$ ms while the maximum value of the network roundtrip time is given by $2T_{net} = 72$ ms.

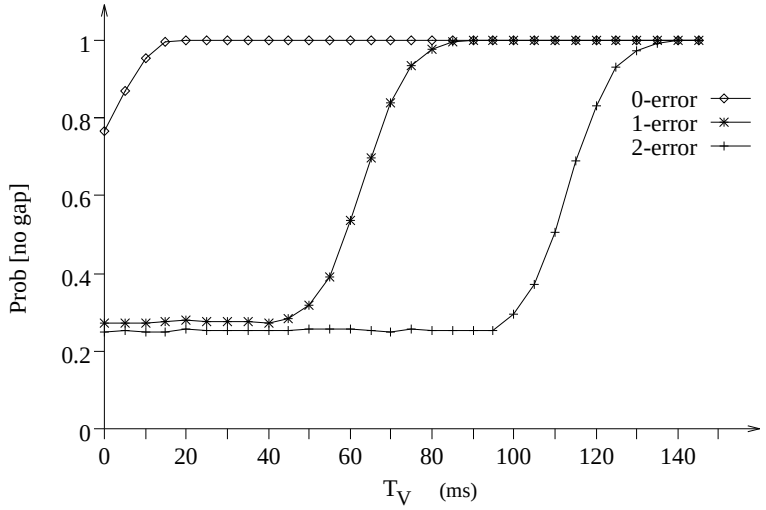

Figure 3.5: Probability of Continuous Playback ($T_{\overline{x}} = 50$ ms).

Therefore, we are guaranteed that a control time of $T_{\overline{x}} + 2T_{net} = 97$ ms allows the successful retransmission of all single packet losses. Yet, in Figure 3.4 we observe that a control time of $T_V = 60$ ms allows us to almost certainly recover from single packet losses, i.e., $Prob[no\ gap] \approx 1$ if $T_V \geq 60$ ms in the *1-error* case. An extension of the argument above to the case of consecutive packet losses shows that each additional consecutive packet lost in an error burst adds approximately the duration of one packetization interval $T_{\overline{x}}$ to the control time T_V required for successful recovery of the entire lost burst. This explains why the retransmission curves in Figures 3.3, 3.4, and 3.5 are parallel with a separation of approximately $T_{\overline{x}}$ ms.

Summarizing, we note that the additional control time needed by Slack ARQ to recover from single packet losses is low for small values of the packetization delay $T_{\overline{x}}$ (Figures 3.3 and 3.4), but may be unacceptable for long packetization delays (Figure 3.5).

3.2.2.2 Experiment 2—Packetization Interval

In this experiment we investigate the effects of different packetization intervals $T_{\overline{x}}$ on Slack ARQ. We assume a fixed control time at the PVR queue, $T_V = 50$ ms. Other simulation parameters are as given in Table 3.1. The results of this experiment are summarized in Figure 3.6. Recall from the previous experiment that since a control time of $T_V = 50$ ms is higher than that required for jitter control in an error-free environment, the *0-error* scenario always yields $Prob[no\ gap] = 1$.

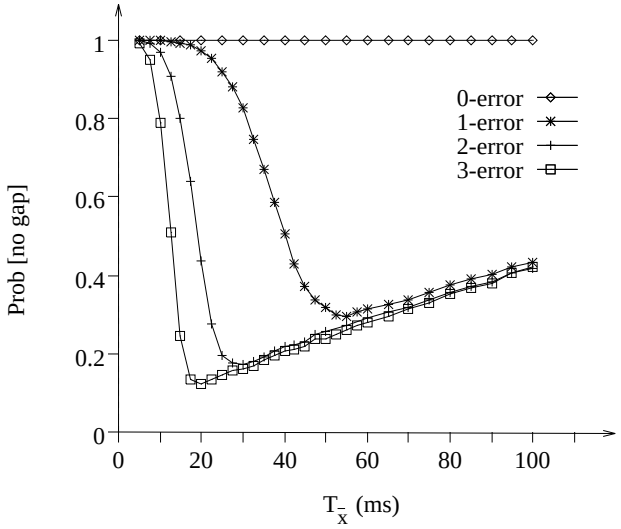

Figure 3.6: Probability of Continuous Playback ($T_V = 50$ ms).

In Figure 3.6 we note the non-monotonic behavior of curves for successful error recoveries using Slack ARQ. Two opposing phenomena are influencing the behavior of the curves. On the one hand, as $T_{\overline{x}}$ grows, the probability of successful retransmissions is lessened. However, as $T_{\overline{x}}$ is increased, the number of packets in a talkspurt decreases and fewer packets per talkspurt increase the likelihood that the lost packet is the first packet in the talkspurt. The simulation model assumes that the first packet in a talkspurt can always be recovered since the playback schedule can still be changed until the first packet is played back. Since network roundtrip times in a local area network are small relative to end-to-end delay constraints for speech quality, retransmission of the first packet is always preferred and always successful.

Distribution	T_{access}	T_{sndr}, T_{rcvr}	
		C_{const}	C_{var}
Exponential	$\Phi(8, 1.00, 16)$	1.0	$\Phi(8, 1.00, 24)$
Erlangian	$\Phi(8, 0.25, 16)$	1.0	$\Phi(8, 0.25, 24)$
Hyperexponential	$\Phi(8, 1.50, 16)$	1.0	$\Phi(8, 3.00, 32)$

Table 3.2: Delay Parameters in Experiment 3 (in milliseconds).

3.2.2.3 Experiment 3—Network Delay

In this experiment we investigate the effects of different distributions for the network delay T_{net} on the control times required for jitter and error control. The packetization time is fixed at $T_{\overline{x}} = 25$ ms. Three different network delay scenarios are investigated, and the parameters in these three scenarios are given in Table 3.2. Recall that $\Phi(x_1, x_2, x_3)$ denotes the truncated Cox distribution with mean x_1, coefficient of variation x_2, and maximum x_3.

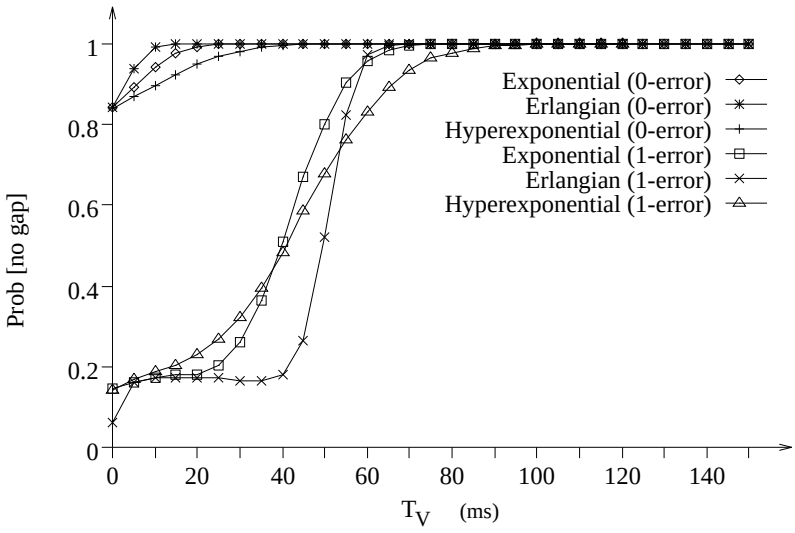

Figure 3.7: Probability of Continuous Playback for Different Distributions for T_{net}.

The curves in Figure 3.7 represent the *0-error* and *1-error* curves for the three T_{net} delay distributions. The *0-error* curves in Figure 3.7 show that higher variation in T_{net}

creates the need for a longer control time to handle jitter. The hyperexponential scenario, for example, requires 35 ms for jitter control while the Erlangian scenario requires only 15 ms. The *1-error* curves show differences as well in the control times required for full coverage of single packet losses.

As the control time increases, the convergence toward full coverage is quite dependent on the delay distribution of T_{net}. The Erlangian scenario exhibits a convergence behavior approximating a discrete jump as it approaches the control time that allows for successful recovery of a single packet. Thus, for a control time of 40 ms, error recovery would be very poor, while with 60 ms almost all lost packets are recovered. The high variation represented by the hyperexponential scenario allows for better recovery at lower control times than the other scenarios, but its convergence to full coverage takes longer.

While the distributions here represent a wide range of variation in the behavior of T_{net}, note that the overall effects are small. Single packet error recovery can be covered for all the distributions with a conservative control time $T_V = 85$ ms and 80 percent coverage is provided with a control time of $T_V = 55$ ms.

3.3 Conclusions

Slack ARQ represents a new unified approach towards the delay and error constraints that influence the quality of the distribution of a digitized voice channel over a packet-switched network. For realistic packet voice transmission scenarios over a loaded local area network, our simulation study indicates that Slack ARQ is a feasible approach to error control. An extended control time for error recovery through retransmission can provide significant error coverage while remaining within the same order of magnitude as the control time required for jitter control.

In the next chapter we develop an analytical end-to-end model for continuous media stream transmission, similar to the simulation model, in order to evaluate the Slack ARQ concept in more general network settings.

Chapter 4

An Analytical End-to-End Model for Slack ARQ

In order to evaluate our retransmission-based error control approach more thoroughly, we develop in this chapter an analytical end-to-end model for the distribution of continuous media traffic. While similar to the simulation model in Chapter 3, the analytical end-to-end model provides a general abstraction for investigating the important protocol and network parameters that drive the dynamic behavior of the playback queue at the receiver and hence determine the effectiveness of delay-constrained retransmission under the Slack ARQ scheme. Besides generality, the analytical approach offers much faster computation, given that the simulation study in Chapter 3 requires hours of computer time on a contemporary compute server in order to produce a single curve for one configuration of parameters.

In Section 4.1 we develop an analytical model for end-to-end transmission of real-time streams and note the differences between it and the simulation model in Chapter 3. In Section 4.2 we use this model to derive analytic expressions for the probability of continuous playback of a talkspurt in the presence of errors, that is, the performance metric for timely retransmissions under Slack ARQ, as developed in Chapter 3. In Section 4.3 we present numerical examples of different end-to-end transmission scenarios to determine parameter sensitivities. In Section 4.4 we summarize the conclusions of this study.

4.1 End-to-End Model

In this section we give a detailed description of the end-to-end retransmission model. The model considers all of the protocol issues for transmission quality of continuous media streams, as presented in Section 1.1 of Chapter 1.

We begin by noting the differences between the analytical end-to-end model and the the simulation model of Chapter 3. The simulation model executed under the assumption of a variable (i.e., exponentially distributed) number of milliseconds in each talkspurt with a fixed packetization interval. The analytical model assumes a fixed number of packets in a talkspurt. The simulation model had a slightly more complex and LAN-specific network delay model. The network delay in the simulation uses truncated distributions to reflect the properties of the FDDI media access protocol and related endsystem behavior. The analytical abstraction models all end-to-end network delays as a single delay variable. Both models take into account reordering effects. Finally, when calculating the performance metric, we assumed in the simulation model that, if lost, the first packet in each talkspurt could always be recovered through retransmission, a realistic assumption in the LAN environment of Chapter 3. As discussed in Section 4.2.2, in the analysis here we exclude consideration of the loss of the first packet in a talkspurt.

The analytical model specifically addresses packet voice transmission. Since packets in different talkspurts rarely interfere with each other, the traffic modeled is the transmission of a single talkspurt within a single packet voice stream. Starting at time $t = 0$, the sender generates voice packets after packetization intervals of length $\overline{x}$. A talkspurt is assumed to consist of a fixed number of N packets. The transmission time of packets is assumed to be negligible compared to the packetization interval. Thus, $\overline{x}$ also represents the distance between packets at the entrance of the network.

The network delay experienced by a packet is described by a distribution F_D. Packet delays are assumed to be independent, and D_j is used to denote the network delay of the jth packet in a talkspurt. However, we enforce in-sequence delivery of voice packets by the network. Thus, a packet with a long network delay may cause a number of packets to arrive

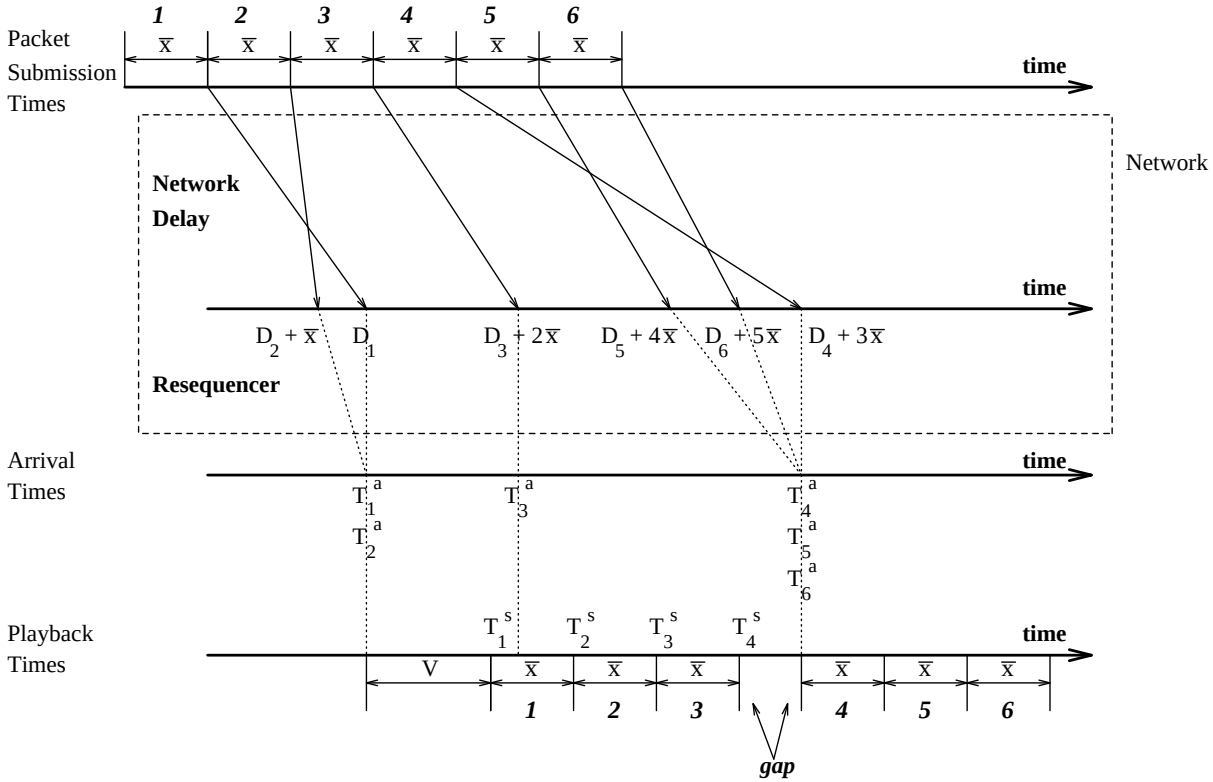

Figure 4.1: Transmission Model of a Talkspurt.

back-to-back at the receiver.

When the first packet of a talkspurt arrives to the receiver, playback of the talkspurt is delayed for the duration of the control time, denoted by V ($V > 0$). Thus, playback of the talkspurt is started at $t = D_1 + V$. The playback duration of each packet is identical to the packetization interval $\overline{x}$.

The end-to-end transmission model is summarized in Figure 4.1 for a talkspurt consisting of $N = 6$ packets. In the top of the figure we show the transmission of packets with a distance of $\overline{x}$ time units. Each packet experiences an independent network delay of D_j for the jth packet of the talkspurt. In-sequence delivery of packets is enforced by the so-called *resequencer* shown in Figure 4.1. Denoting the arrival time of the jth packet by T_j^a, the resequencer enforces that

$$T_j^a = \max_{i=1,\ldots,j} \{D_i + (i-1)\overline{x}\} \tag{4.1}$$

Thus, packets arrive to the receiver in the order in which they were transmitted. Assuming that no playback discontinuities have occurred before the arrival of the jth packet, the scheduled playback time for the jth packet, denoted by T_j^s, is fully determined once the first packet has arrived at the receiver. T_j^s is given by

$$T_j^s = D_1 + V + (j-1)\overline{x} \tag{4.2}$$

The first packet in a talkspurt that arrives after its playback time, that is, $T_j^s < T_j^a$, causes a discontinuity, the so-called *gap*, in the playback of the talkspurt. In Figure 4.1, a gap can be observed before the fourth packet.

The error scenario for our end-to-end model specifies that during the transmission of a talkspurt there is an arbitrary period during which zero or more packets are dropped by the network, the so-called *error period*. We assume only one error period per talkspurt, but allow multiple consecutive packets to be dropped in the error period. Since packets of a talkspurt arrive at the receiver in the order in which they were transmitted, the receiver detects losses as soon as a packet arrives out-of-sequence. After detecting a loss, a retransmission procedure is initiated. We assume that the time to recover lost packets via retransmission is fully determined by a roundtrip network delay. Processing times for retransmissions at the receiver or the sender are assumed to be small and not considered in our model. Also, we assume that the sequence of lost packets in the error period can be retransmitted in a single packet. This assumption is realistic for network or transport layer protocols since the maximum packet size is much larger than the typical voice packet. Thus, denoting the retransmission time by R, the distribution function of R is given by

$$F_R = F_D \otimes F_D$$

where $\otimes$ is the convolution operator.

4.2 Analysis of End-to-End Model

In this subsection we develop an analytic expression for the probability of continuous playback for a talkspurt. First, we derive the desired probability assuming an error-free scenario.

Then, we extend our expression to consider error periods in a talkspurt.

4.2.1 Probability of Continuous Playback Without Errors

We are concerned with the occurrence of a gap in the playback of a talkspurt. We thus define random variables G_i $(1 \leq i \leq N)$ that can be used to indicate the presence of a discontinuity in the playback. By setting

$$G_i := \begin{cases} V & \text{if } i = 1 \\ 0 & \text{if } G_{i-1} = 0,\ i \neq 1 \\ \max\{0, T_i^s - T_i^a\} & \text{otherwise} \end{cases} \tag{4.3}$$

we obtain $G_i = 0$ if a packet with index i or less arrives after its playback time. Since the arrival of the first packet sets the playback schedule and cannot cause a gap, that is, $T_1^s - T_1^a = V$, we set $G_1 = V$. For a talkspurt with N packets, $G_N > 0$ indicates that no discontinuity has occurred during playback of the entire talkspurt.

Note that in equation (4.3), $G_i = 0$ for the ith packet is feasible in two scenarios. Either no gap has occurred before packet i and packet i arrives after its playback time, or a gap has occurred before the arrival of packet i. Therefore, denoting by $\mathrm{P}\{G_i = 0\}$ the probability of $G_i = 0$, we obtain for $2 \leq i \leq N$

$$\mathrm{P}\{G_i = 0\} \quad = \quad \mathrm{P}\{G_{i-1} = 0\} + \mathrm{P}\{G_{i-1} > 0 \ \textit{and}\ T_i^s < T_i^a\} \tag{4.4}$$

Note that the second term on the right side of equation (4.4) is the probability that the ith packet causes the first gap in the playback of the talkspurt. In this case, all packets with index less than i have arrived before their respective playback times. It follows that packet i could not have arrived earlier than any packet with a smaller index, that is,

$$T_i^a = \max_{j=1,\ldots,i}\{D_j + (j-1)\overline{x}\} = D_i + (i-1)\overline{x} \tag{4.5}$$

Substituting equation (4.5) into equation (4.4) yields

$$\mathrm{P}\ \{G_i = 0\} \quad = \quad \mathrm{P}\ \{G_{i-1} = 0\} + \ \mathrm{P}\ \{G_{i-1} > 0 \ \textit{and}\ V + D_1 < D_i\} \tag{4.6}$$

We can now recursively compute the probability for continuous playback, i.e., $G_N > 0$, by

$$\mathrm{P}\ \{G_N > 0\} \quad = \quad 1 - \int_0^\infty \mathrm{P}\{G_N = 0 \mid D_1 = t\} dF_D(t) \tag{4.7}$$

4.2.2 Probability of Continuous Playback in the Presence of Errors

In the presence of errors, gaps in the playback of a talkspurt may result from delay jitter or from a failure of the retransmission procedure to recover lost packets before their playback times. In this subsection we calculate the probability of continuous playback of a talkspurt, given that the network loses k consecutive packets, say packets with index $n-k, n-k+1, \ldots, n-1$. We exclude the loss of the first packets in a talkspurt, i.e., $n > k+1$. Note that in a talkspurt containing an error period, the arrival time of the jth packet is obtained by calculating the latest packet arrival with index less than j that is not lost in the network. Thus we obtain for T_j^a that

$$T_j^a = \begin{cases} \max\limits_{l=1,\ldots,j} \{D_l + (l-1)\overline{x}\} & \text{if } j \leq n-k-1 \\ \max\limits_{l=1,\ldots,n-k-1,n,n+1,\ldots,j} \{D_l + (l-1)\overline{x}\} & \text{otherwise} \end{cases} \tag{4.8}$$

For calculating the probability of continuous playback in the presence of errors, we must consider two cases which can result in gaps. First, a gap may be due to an untimely retransmission of the lost packets. We refer to this case as an *error gap* or *E-gap*. Second, a gap may result from excessive delay variations, independent of the lost packets. This case is referred to as a *jitter gap* or *J-gap*. Thus, $\mathrm{P}\{gap\}$, the probability of a gap in the playback of a talkspurt, is given by

$$\mathrm{P}\{gap\} \quad = \quad \mathrm{P}\{E\text{-}gap\} + \mathrm{P}\{no\ E\text{-}gap\}\mathrm{P}\{J\text{-}gap \mid no\ E\text{-}gap\} \tag{4.9}$$

The calculation of $\mathrm{P}\{gap\}$ is performed in three steps. We first calculate the probability of an *E-gap*. Next, using an approach similar to that of the previous subsection, we define random variables G_i, that indicate the occurrence of a *J-gap* at or before the arrival of packet i. We then calculate $\mathrm{P}\{J\text{-}gap \mid no\ E\text{-}gap\}$, which is the probability of a *J-gap* in the talkspurt under the condition that the lost packets are retransmitted in a timely fashion.

Note that a jitter gap can occur before or after the lost packets are due for playback. In the latter case, the discontinuity occurs independent of the retransmission procedure.

First we consider the probability of gaps due to untimely retransmission. Recall that the time necessary for retransmission is denoted by R with $F_R = F_D \otimes F_D$ and the arrival of the nth packet, that is, the first packet after the sequence of lost packets, invokes the retransmission procedure. For retransmission to be untimely, the difference between the playback time of the $(n-k-1)$st packet, i.e., the first lost packet, and the arrival time of the nth packet must be less than R, the time necessary for retransmission. Thus, the probability of an *E-gap* is

$$\begin{aligned} \mathrm{P}\{\textit{E-gap}\} &= \mathrm{P}\{T^s_{n-k} - T^a_n < R\} \\ &= \mathrm{P}\left\{ \max_{j=1,\ldots,n-k-1,n} \{D_j + (j-1)\overline{x}\} > V + D_1 + (n-k-1)\overline{x} - R \right\} \end{aligned} \tag{4.10}$$

Fixing the retransmission time to $R \equiv r$, and the delay of the first packet to $D_1 \equiv t$, we obtain from equation (4.10) that

$$\mathrm{P}\left\{ \max_{j=2,..,n-k-1,n} \{t, D_j + (j-1)\overline{x}\} > (V + t + (n-k-1)\overline{x} - r) \mid D_1 = t,\ R = r \right\} = \tag{4.11}$$

$$1 - \left(\mathrm{P}\left\{V + (n-k-1)\overline{x} > r\right\} F_D(V + t - k\overline{x} - r) \prod_{j=2}^{n-k-1} F_D(V + t + (n-k-j)\overline{x} - r) \right)$$

In equation (4.11), we have used the independence of network delays. Now we uncondition the expression in equation (4.11) through integration over the retransmission delay and the delay of the first packet in the talkspurt. That is,

$$\begin{aligned} \mathrm{P}\{\textit{E-gap}\} = \int_t \int_r 1 - &\ \mathrm{P}\ \left\{V + (n-k-1)\overline{x} > r\right\} F_D(V + t - k\overline{x} - r) \cdot \\ &\left(\prod_{j=2}^{n-k-1} F_D(V + t + (n-k-j)\overline{x} - r) \right) dF_R(r) dF_D(t) \end{aligned} \tag{4.12}$$

$$= 1 - \int_t \int_{r=0}^{V+(n-k-1)\overline{x}} F_D(V + t - k\overline{x} - r) \prod_{j=2}^{n-k-1} F_D(V + t + (n-k-j)\overline{x} - r) dF_R(r) dF_D(t)$$

In the following we assume that retransmission is timely, that is, an *E-gap* has not occurred. We compute P{*J-gap* | *no E-gap*}, the probability that a *J-gap* occurs in the talkspurt under the assumption of timely retransmission. Similarly to equation (4.3), we define random variables G_i, such that $G_i = 0$ if a packet with index i or less arrives after its playback time. With our assumption that packets $n-k, n-k+1, \ldots, n-1$ are lost and do not arrive to the receiver we obtain for $1 \leq i \leq n-k-1$ or $n \leq i \leq N$

$$G_i := \begin{cases} V & \text{if } i = 1 \\ G_{n-k-1} & \text{if } i = n \\ 0 & \text{if } G_{i-1} = 0, i \neq 1, i \neq n \\ \max\{0, T_i^s - T_i^a\} & \text{otherwise} \end{cases} \tag{4.13}$$

We are interested in $\mathrm{P}\{G_N = 0\}$. Since the definition of G_i is recursive, we must compute $\mathrm{P}\{G_i = 0\}$ for all values of i. From the definition in equation (4.13), $\mathrm{P}\{G_1 = 0\} = 0$. That is, the first packet sets the playback schedule and cannot cause a jitter gap. Also, the nth packet cannot cause a jitter gap since its arrival invokes the retransmission procedure, which could not be timely if the nth packet arrived after its playback time. Hence $\mathrm{P}\{G_n = 0\}$ is defined to be the probability that a jitter gap occurs before the nth packet, i.e., $\mathrm{P}\{G_n = 0\} = \mathrm{P}\{G_{n-k-1} = 0\}$. It remains to consider two cases: $2 \leq i \leq n-k-1$ and $n \leq i \leq N$.

We first calculate the probability of a *J-gap* at or before the ith packet where $i \leq n-k-1$. Since no *E-gap* occurs, the ith packet arrives early enough to ensure successful retransmission of the lost packets, but the ith packet can cause a jitter gap by arriving after its playback time. That is, the conditions for the ith packet resulting in a jitter gap are

$$T_i^a > T_i^s \qquad \text{and} \qquad T_i^a \leq T_n^a < T_{n-k}^s - R \tag{4.14}$$

Hence, for $2 \leq i \leq n-k-1$,

$$\mathrm{P}\{G_i = 0\} \;=\; \mathrm{P}\{G_{i-1} = 0\} + \mathrm{P}\{G_{i-1} > 0 \;\; and \;\; T_i^s \leq T_i^a \leq T_{n-k}^s - R\} \tag{4.15}$$

The last term in equation (4.15) represents the probability of the first jitter gap in the playback occurring at the ith packet. Following the argument for equation (4.5), the arrival time of the ith packet is $D_i + (i-1)\overline{x}$, and equation (4.15) can be rewritten as

$$\begin{aligned} \mathrm{P}\{G_i = 0\} &= \mathrm{P}\{G_{i-1} = 0\} \\ &+ \mathrm{P}\{G_{i-1} > 0 \ \ and \ \ V + D_1 \leq D_i \leq V + D_1 + (n-k-i)\overline{x} - R\} \end{aligned} \tag{4.16}$$

Equation (4.16) is equivalent to

$$\begin{aligned} \mathrm{P}\{G_i = 0\} = &\int_t \int_r \mathrm{P}\{G_{i-1} = 0 \mid D_1 = t, R = r\} dF_R(r) dF_D(t) \ + \\ &\int_t \int_r \mathrm{P}\{G_{i-1} > 0 \mid D_1 = t, R = r\} (F_D(V + t + (n-k-i)\overline{x} - r) - F_D(V + t)) dF_R(r) dF_D(t) \end{aligned} \tag{4.17}$$

We have shown how to compute $\mathrm{P}\{G_i = 0\}$ for $i \leq n-k-1$. Next we consider $\mathrm{P}\{G_i = 0\}$ for $i > n$. Since packets with index greater than n cannot affect the retransmission procedure, we obtain with the definition in equation (4.13):

$$\mathrm{P}\{G_i = 0\} \quad = \quad \mathrm{P}\{G_{i-1} = 0\} + \mathrm{P}\{G_{i-1} > 0 \quad and \quad T_i^s \leq T_i^a\} \tag{4.18}$$

By an argument similar to that for equation (4.5), the arrival time of the ith packet in the last term of equation (4.18) is $T_i^a = D_i + (i-1)\overline{x}$. Then, Equation (4.18) yields, for $n+1 \leq i \leq N$,

$$\mathrm{P}\{G_i = 0\} = \int_t \mathrm{P}\{G_{i-1} = 0 \mid D_1 = t\} + \mathrm{P}\{G_{i-1} > 0 \mid D_1 = t\} \cdot \mathrm{P}\{V + t \leq D_i\} dF_D(t) \tag{4.19}$$

We can now compute $\mathrm{P}\{gap\}$, the probability of a gap in the playback of a talkspurt, from equation (4.9) since we have $\mathrm{P}\{E\text{-}gap\}$ in equation (4.13) and $\mathrm{P}\{J\text{-}gap \mid no\ E\text{-}gap\}$ by recursive evaluation of equations (4.17) and (4.19).

4.3 Numerical Examples

In this section we apply our analysis to four network transmission scenarios. In each example the effectiveness of retransmissions is expressed in terms of the probability of maintaining playback continuity during a talkspurt, as derived in the previous section.

Example	Packetization Interval ($\overline{x}$)	N	Avg. Net Delay	Net Distribution (F_D)
1	20 ms	20	15 ms	E_2
2	20 ms	20	15 ms	E_1, E_2, E_6
3	20 ms	20	10,20,30,40 ms	E_2
4	6 ms	60	15 ms	E_2

Table 4.1: Parameters for Numerical Examples.

For the network delay distribution, F_D, we consider delay distributions with different levels of variance, namely Erlang-k distributions, denoted by E_k, for $k \geq 1$. Networks with large delay variations are modeled by E_1, that is, an exponential distribution; for moderate and low delay variations we use, respectively, E_2 and E_6.

The parameters for our examples are presented in Table 4.1 in which the packetization interval $\overline{x}$ and the average network delay are given in milliseconds. Example 1 investigates the sensitivity of the probability of continuous playback in the presence of errors for a base transmission scenario. In the base scenario, we have selected E_2 as the default network delay distribution. The selection reflects that delay variations over short periods of times, such as the duration of a talkspurt, are generally modest. The mean network delay is set to 15 ms. The packetization interval is fixed at $\overline{x} = 20$ ms, a value commonly used in extant voice protocols [50], and each talkspurt consists of $N = 20$ packets. Together with the packetization interval, this corresponds to a talkspurt length of 400 ms, a value motivated by our empirical measurements of packet voice traffic in Chapter 5 (see Figure 5.7). The other examples vary one or more parameters of the base transmission scenario. In Example 2 we show the degree to which retransmission-based error recovery is influenced by the network delay distribution. In Example 3 we consider different mean network delays with the same (e.g., E_2) network delay distribution. In Example 4 we consider the effects of reducing the packetization interval and subsequently increasing the number of packets in a talkspurt. A motivation for Example 4 is the consideration of our retransmission scheme in

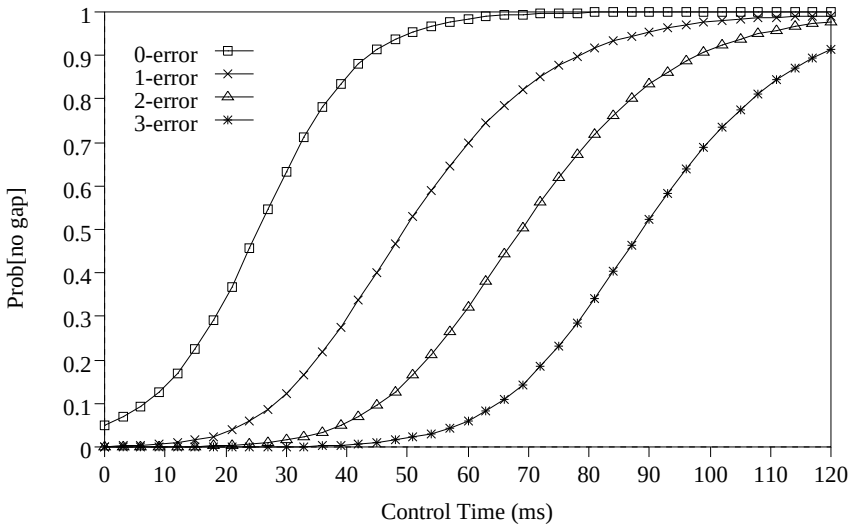

Figure 4.2: Retransmission Effectiveness for an E_2 (Erlang-2) Network Delay Distribution.

an ATM adaptation layer protocol [11], instead of an upper layer protocol.

4.3.1 Example 1: Effects of the Control Time on Retransmission

Recall that in our end-to-end model, we specify a single error period during the transmission of a talkspurt, but multiple consecutive packets can be lost during the error period. Here we consider error periods in which zero, one, two, or three packets are lost. An error period in which i packets are lost is referred to as an *i-error* scenario.

Figure 4.2 shows the probability of continuous playback of the talkspurt under variation of the control time. It has four curves representing the respective error scenarios. The *0-error* scenario gives the probability that delay jitter results in a discontinuity in talkspurts whose end-to-end transmission is error-free. From the *0-error* curve we see that a control time of roughly $V = 60$ ms is required to compensate for the delay jitter in the network. With $V = 60$ ms, approximately 70% of single-packet losses are successfully recovered through retransmission as are 30% of 2-packet losses. As the control time is lengthened, error coverage improves and at $V = 100$ ms, successful retransmission in both the *1-error* and *2-error* scenarios occurs in 90% of the cases, while for the *3-error* scenario it is approximately 70%.

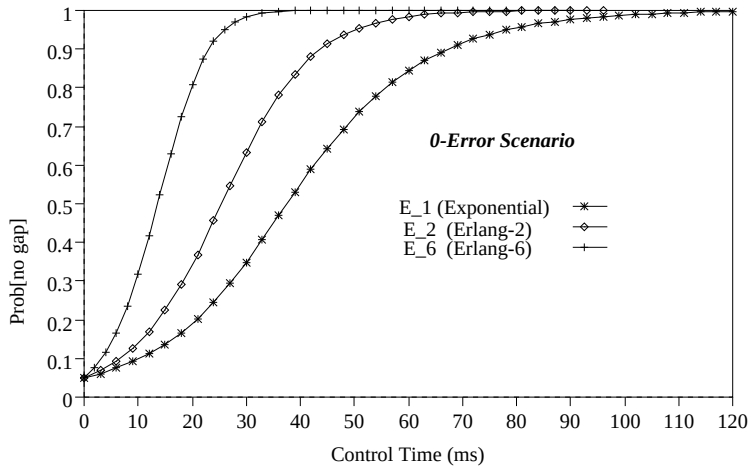

Figure 4.3: Delay Jitter for Different Network Delay Distributions.

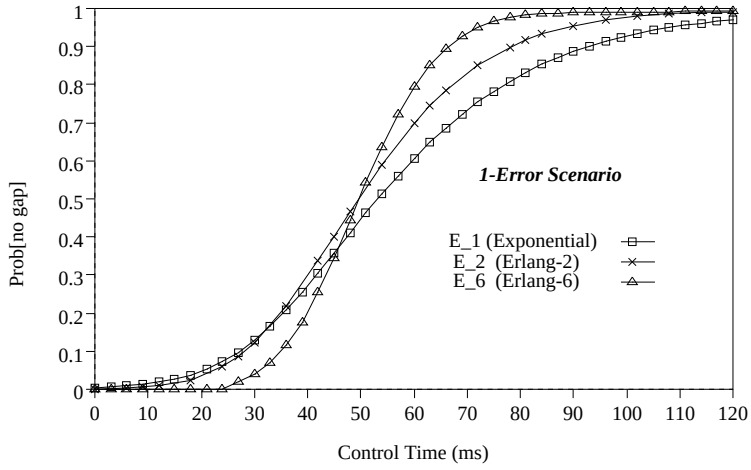

Figure 4.4: Retransmission Effectiveness for Different Network Delay Distributions.

Recall that the feasible range of control time values is determined by the end-to-end delay restriction. In our example, the sum of the packetization interval and the network delay on the average account for only 35 ms of the total end-to-end delay. Control times on the order of $V = 100$ ms are thus feasible for all but the most stringent delay requirements.

The packetization interval plays an important role in the retransmission algorithm. In a k-error scenario, the average amount of time that elapses between the occurrence of a loss in the network and its discovery at the receiver is $k\overline{x}$ since the receiver discovers packet loss when the first out-of-sequence packet arrives. Hence the probability of successful

retransmission in a k-error scenario will be low when $V < k\overline{x}$. This can be observed in Figure 4.2 where the recovery rate for control times of less than $k\overline{x}$ is roughly 5%, e.g., for the *3-error* scenario at a control time of $V = 60$ ms, 6% of retransmission attempts are successful. As with the curves from Experiment 1 in the simulation study in Chapter 3, the influence of the packetization interval is graphically evident in Figure 4.2—to achieve a fixed probability of successful retransmission, the control times required for the *1-error* and *2-error* scenarios differ by approximately the size of the packetization interval. The same relationship is observed between the control times for the *2-error* and *3-error* scenarios.

4.3.2 Example 2: Effects of Network Delay Variation

Here we study the effects of different network delay distributions on the probability of successful retransmission. We select E_1 to represent networks with high delay variations, E_2 for moderate delay variations, and the E_6 for low delay variations. Before examining retransmission behavior, we first consider the differences in the amount of delay jitter in the network under these delay distributions.

Figure 4.3 shows the probability of continuous playback of a talkspurt for the respective network delay distributions, given that the transmission of the talkspurt is error-free. The size of the control times necessary to compensate fully for the delay jitter in the network varies widely— $V = 100$ ms for E_1, but only $V = 60$ ms for E_2, and $V = 30$ ms for E_6.

Figure 4.4 compares the probability of successful retransmission of single-packet losses. For small control times, the higher variation of E_1-distributed delays results in a greater probability of successful retransmission than the other distributions. However, E_1 results in a lower probability of successful retransmission than E_2 and E_6 for larger control time values. As shown in Figure 4.4, the E_1 curve is crossed by the other curves at points where the probability of successful retransmission is low, e.g., at about 0.35 for E_6. Hence the delay distributions with low variation will inherently recover more packets successfully. For instance, when the control time is $V = 70$ ms, the probability of successful retransmission is approximately 0.73 for E_1, 0.85 for E_2, and 0.93 for E_6.

4.3.3 Example 3: Effects of Average Network Delay

In this example we examine the effects of varying the mean network delay for a fixed network delay distribution, i.e. E_2. All other parameters are the same as in Example 1. The *0-error* curves for mean network delays of 2, 10, 20, 30, and 40 ms are shown in Figure 4.5 and the *1-error* curves in Figure 4.6.

The curves in Figure 4.5 indicate that the control time required to compensate fully for delay jitter is approximately four times the mean network delay under the parameters of this example. Note that scenarios with mean network delays larger than 40 ms are marginally feasible in this network, due to the interaction between buffering for jitter and the end-to-end delay constraint. If the mean network delay is 40 ms and the end-to-end delay bound is 200 ms, then the maximum feasible control time is approximately $200 - 20 - 40 = 140$ ms whereas the control time required to compensate fully for delay jitter is approximately 160 ms.

Figure 4.6 shows the effect on retransmission effectiveness of increasing the mean network delay. Retransmissions are rarely successful whenever the control time is less than 20 ms since, as discussed in Example 1, the packetization interval represents the time required for the discovery of a single-packet loss at the receiver. The curve for a mean network delay of 2 ms graphically illustrates this lower bound on the control time required for successful retransmissions.

The curves reveal a strong interaction between retransmission effectiveness and the one-way network delay. As the average network delay increases so does the absolute size of delay jitter in the network. Retransmission behavior is also linked to the roundtrip network delay, amplifying the effects of increases in one-way network delays. Consider an 80 ms control time in Figure 4.6. If the average network delay is 10 ms, this control time will allow for the recovery of 100% of single-packet losses. If the average network delay is 20 ms, coverage falls to 80%, for 30 ms to approximately 50%, and for 40 ms to approximately 25%.

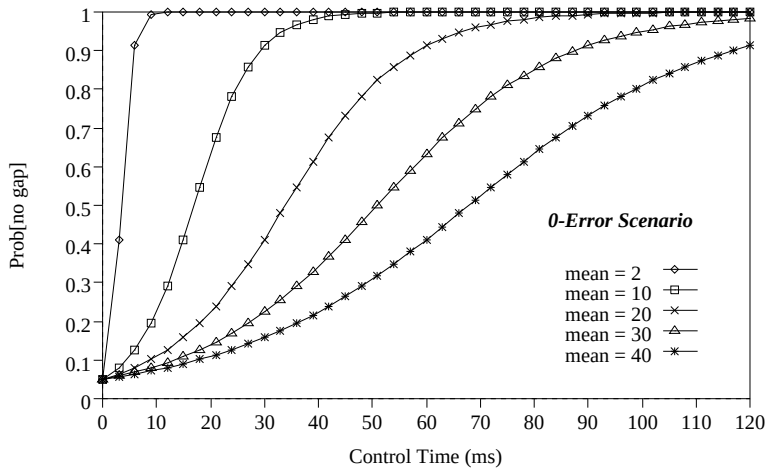

Figure 4.5: Delay Jitter for Different Mean Network Delays.

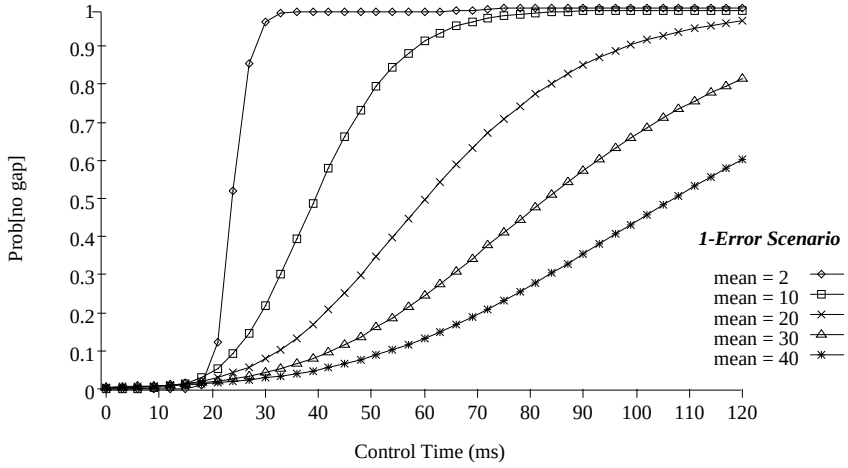

Figure 4.6: Retransmission Effectiveness for Different Mean Network Delays.

In summary, for the network scenario in Example 3, we conclude that our retransmission-based approach to error control will not be advantageous if the mean network delay is at or above roughly 30 ms. For mean network delays below 30 ms, the results suggest that there will be significant error coverage for small burst losses when using the retransmission scheme.

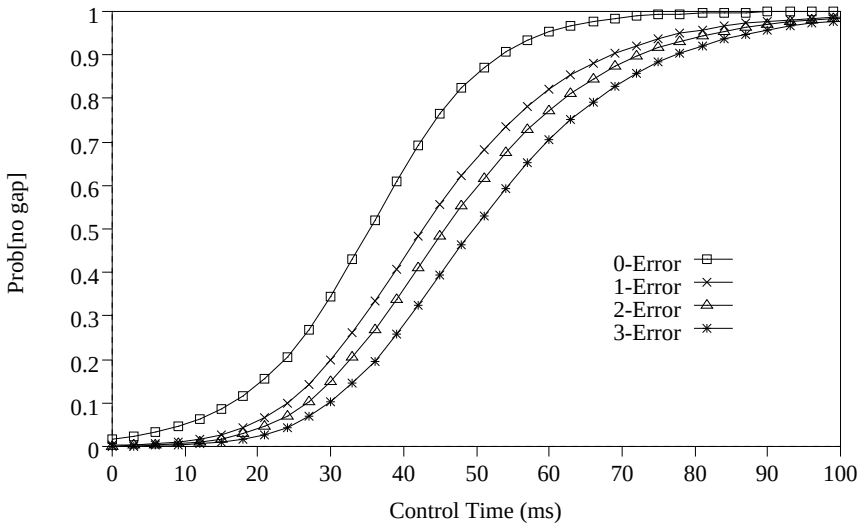

Figure 4.7: Retransmission Effectiveness for an ATM Cell-Level Protocol.

4.3.4 Example 4: Effects of Packetization Interval

Example 4 investigates the effects of reducing the packetization interval and subsequently increasing the number of packets in each talkspurt. We choose a packetization interval of 6 ms and $N = 66$ packets in each talkspurt. The duration of a talkspurt is thus approximately the same as in Example 1. The choice of parameters is motivated by consideration of Slack ARQ being embedded into a lower layer protocol in an ATM network, i.e., a connection-oriented service at the ATM adaptation layer. The packetization interval corresponds to uncompressed PCM-encoded voice carried in a 48-byte ATM cell payload.

We observe from Figure 4.7 that the *0-error* curve rises much slower than the corresponding *0-error* curve in Example 1 (see Figure 4.2). Both curves, however, show that the control time to fully compensate for delay jitter is 70 ms. The elongated curve in Figure 4.7 reflects the increased probability of a jitter gap since each talkspurt consists of $N = 66$, as opposed to $N = 20$, packets. Note that the increase in the control time is relatively modest, given that each talkspurt has over three times as many packets. In Example 1, for instance, the control time required to reach a 0.9 probability of continuous playback is 46 ms; in Example 4 it is 52 ms. The implication is that the performance metric is relatively insensitive to selection of parameter N.

Decreasing the packet size dramatically improves the error coverage provided by our retransmission scheme. As discussed in Example 1, the "knee" of the *k-error* curve is found at approximately k$\overline{x}$. Figure 4.7 indicates that recovery of burst losses containing several packets (cells) is feasible under this example, whereas the recovery of large bursts was not feasible in Example 1, i.e., with $\overline{x} = 20$.

4.4 Conclusions

We have employed analytical modeling techniques to determine the effectiveness of delay-constrained retransmission for different transmission scenarios. The analysis enables us to quantify the interactions between network delay and protocol-controlled parameters. We summarize our findings as follows:

- The analytical results support the simulation study of Chapter 3 in demonstrating the feasibility of delay-constrained retransmission for many realistic transmission scenarios.

- For a fixed average network delay, the sensitivity of the retransmission scheme to changes in the variability of network delays is not strong. The *k-error* retransmission curves in our figures display a characteristic S-shape. Less delay jitter causes the curve to rise more rapidly whereas more delay jitter tends to flatten the curve. Relative to the range of feasible control times, retransmission success was not observed to vary greatly with varying delay jitter in the network.

- Retransmission effectiveness is closely tied to the average one-way network delay. Small increases in the average network delay result in proportionally larger increases in network jitter and roundtrip delays, which reduces retransmission effectiveness. One important network characteristic is thus the rate at which increasing absolute network delays will produce larger absolute variations in network delays.

- The packetization interval plays a key role in the retransmission process, primarily due to the assumption that the discovery of a packet loss at the receiver is based on

the arrival of out-of-order packets. From a practical viewpoint, we believe that an implementation of the Slack ARQ scheme must use gap-based error detection. Timer-based detection is less responsive, and timers are notoriously difficult to tune. The implication of gap-based detection is that control times for effective retransmission of a *k-error* loss must be at least k$\overline{x}$ in duration.

As shown in Example 4, reduction of the packetization interval has a dramatic effect on the effectiveness of retransmission for recovery of multiple-packet burst losses. Protocols can influence the retransmission behavior through the selection of small packet sizes. As hosts become more powerful and operating systems provide finer granularity support for timers, higher layer protocols will be more likely to afford small packet sizes.[1] In particular we have considered the effect of packetization at the level of an ATM cell.

To compare our theoretical results with the dynamic behavior of real network environments, in the next chapter we present empirical delay measurements of voice transmissions over a large campus-wide network.

[1] The popularity of 20 ms as the packetization interval in current packet voice software packages is in part driven by the timing characteristics of current audio hardware, which in turn is derived from the timing support available from contemporary operating systems.

Chapter 5

An Empirical Study of Packet Voice Distribution over a Campus-Wide Network

In this chapter we empirically investigate packet voice distribution over a contemporary campus-wide network. In Section 5.1 we outline the purposes of the study. In Section 5.2 we describe the experimental environment and present the empirical data collected on delays in packet voice transmissions. In Section 5.3 we analyze the data and consider its implications for our work. We summarize our results in Section 5.4.

5.1 Goals of the Study

Our primary goal in this study is to provide insight into the extent to which the end-to-end model of Chapter 4 captures the behavior of voice transmissions in a real network. The empirical measurements allow us to test, at least for one network environment, certain underlying assumptions in the analytical model and to compare empirically derived statistics with the retransmission behaviors predicted in our analytical studies. Understanding network delay characteristics is also a first step towards consideration of protocol mechanisms that would best implement the Slack ARQ scheme.

A secondary goal of this study is to examine the potential of extant large LANs for supporting continuous media traffic. The network we study, which is the University of Virginia campus-wide network, is representative of contemporary large enterprise networks

consisting of Ethernet segments connected by high-performance routers to high-speed backbones, e.g., FDDI rings. These networks are good candidates for near-term deployment of continuous media applications. Unlike a small LAN, a campus-wide network is geographically distributed over a large area, making conferencing applications attractive. Unlike most wide-area networks, a campus-wide network has high-bandwidth links and powerful routers and therefore might be expected to have characteristics similar to small LANs. The campus-wide network environment, however, was not designed with the delay sensitivities of continuous media streams in mind, and current applications are largely insensitive to delays at the timing granularity of our measurements.

Emerging high-speed multiservice networks will increase the bandwidth available for continuous media and multimedia applications, and provide quality of service mechanisms to aid in transporting delay-sensitive packet streams. In large enterprise networks these technologies will first appear in backbone networks and isolated local area networks since most enterprises will transition slowly from their current network infrastructure. We believe, however, that continuous media applications will become widely available during this transitional period, and that there will be considerable interest in running continuous media applications over enterprise-wide networks whose component LANs represent the current networking technologies. Our study provides a contribution towards understanding how well this type of network will support continuous media streams, without tuning or modifications to the network.

Measurement studies to characterize large networks remain something of a black art since the enormous number of potential communication pairs renders any systematic probing of a large network infeasible, and since there are different measures of performance. Generally, researchers have focused on roundtrip delays and packet losses, either to understand dynamic behavior for a particular protocol, e.g., TCP [34, 41], or for characterization of aggregate network traffic [44, 48]. Of relevance to our study is the work in [48], in which wide-area network delays are measured using small UDP packets sent every 39 ms from a source to a destination node. These frequent packet probes are used to detect anomalous behavior in the Internet. The timescale of the network delays observed is on the same order

of magnitude as that in our work, and our results suggest the existence of phenomena in the University of Virginia network similar to those reported in [48].

Our focus, however, is the transport of continuous media, and specifically packet voice, over large multiple-segment LANs. No previous empirical studies of continuous media have been done in this environment, and little work has been done on the measurement of real-time traffic in current packet-switched networks. We obtain two types of measurements. First, we measure the characteristics of a 5-minute packet voice stream created with contemporary workstation-based audioconferencing software. Second, we measure the distortions due to network delay that occur during transmissions of this voice traffic, when carried in UDP datagrams, across different paths in the campus-wide network. These experiments allow comparison of empirical statistics with the retransmission probabilities obtained in our simulation and analytical studies.

5.2 Packet Voice Experiments

5.2.1 Experimental Approach

The network for our experiments is the University of Virginia campus-wide network, shown in Figure 5.1. Our measurements were taken over network paths that consist of local Ethernet segments connected by high-performance routers (i.e., Cisco AGS+) to high-speed backbone networks (i.e., FDDI rings). Our approach was to measure the performance over a set of different paths in the network from a single traffic-generating node, located on subnetwork 60, which is attached to the router labeled *Olsson* in Figure 5.1. We selected three destination machines, which are desktop workstations, located on different subnetworks throughout the network. The three paths from the source node to the destination machines explore a significant portion of the network topology.

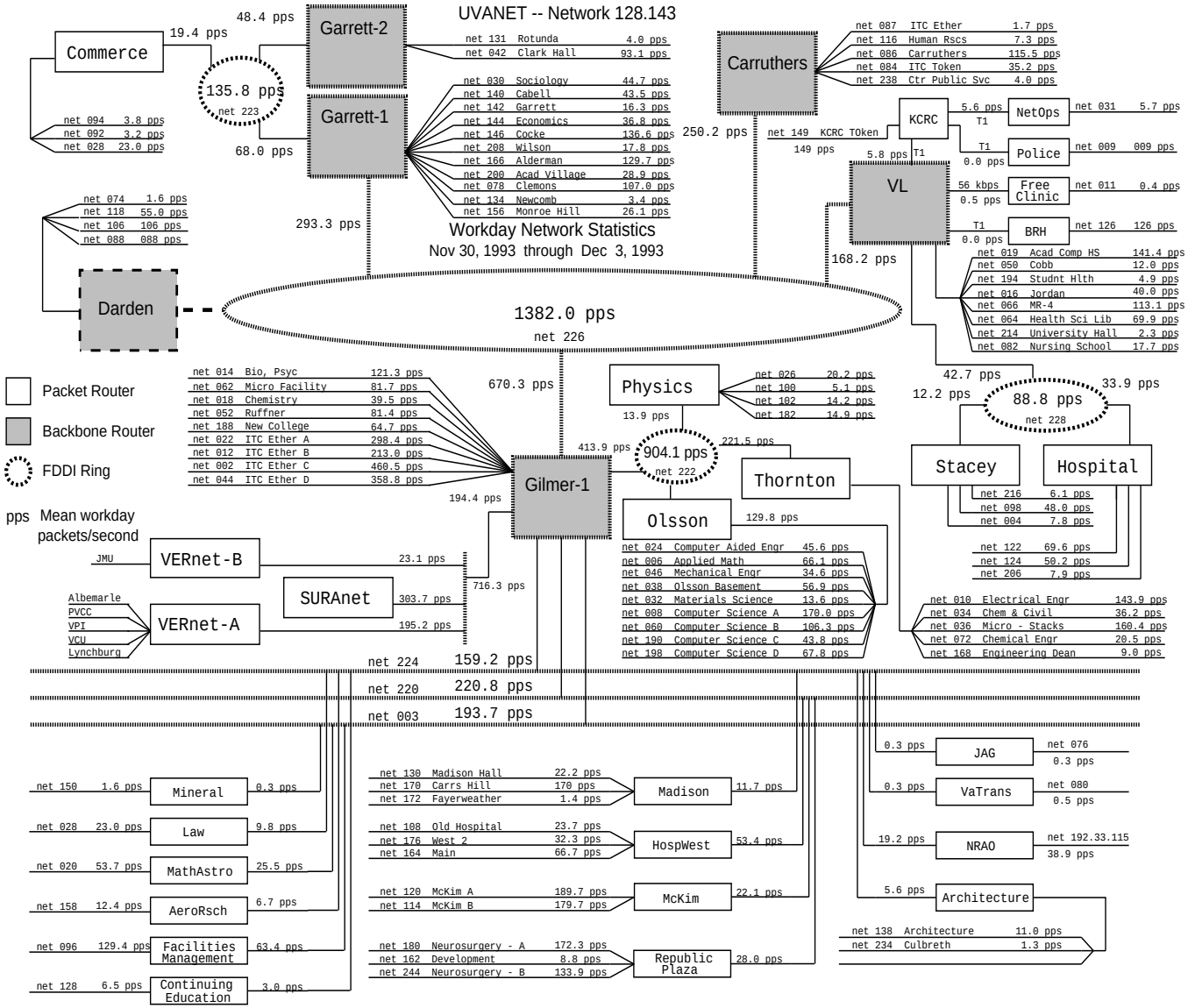

Figure 5.1: University of Virginia Campus Network.

Our goal was to observe the conditions under which a typical continuous media application would run. Therefore, measurements were taken during the daytime, and no attempt was made to regulate the use of the remote machines during the experiments, though we observed very light usage. To control time-dependent behavior, experiments using the different paths were performed back-to-back.

Our work focused on two aspects of packet voice transmission: the traffic profile generated by actual packet voice transmissions and the network delays experienced during voice transmissions across the campus-wide network. In order to have tight control over network delay measurements, we first captured the network traffic profile of a packet voice stream, i.e., the interpacket transmission times and packet sizes of a voice stream transmitted on the network by an audioconferencing application. We then collected network delay statistics on a packet stream transmitted according to the voice traffic profile.

5.2.2 Software Tools

To obtain a network traffic profile for packet voice, we used the voice transmission capability of the INRIA Videoconferencing System (IVS), a public domain software videoconferencing tool [59], running on a Sun SPARCstation. We selected 32 kbit/s ADPCM encoding with silence detection and the default packetization interval of 20 ms. The voice transmitted was monologue male speech.

Packet delays were measured by a software-based network monitoring tool [40] running on an Intel 486 workstation, which was attached to the same local-area network as the traffic-generating workstation. This network monitor node captured each packet on the local network segment and timestamped the packet with a clock accuracy of ± 1 ms. To calculate roundtrip delays, packets were timestamped as they left and as they returned to the local network segment. Examination of sequence numbers in packet headers verified that the network monitor did not drop packets.

5.2.3 Measurements

To obtain a traffic profile of voice transmissions we first transmitted approximately 5 minutes of male monologue speech using the IVS package and used the network monitor to determine the size and interpacket times in this packet stream. The voice processing in the IVS software does not mark talkspurt boundaries, which allows us to capture the audio packet stream without distortions due to protocol-specific processing. [1]

Ideally, we would like to measure the *network delay* experienced by each packet. The network delay is defined to be the time between when the voice protocol software at the source makes a system call to transmit a datagram (i.e., a voice packet) and when the voice protocol software at the receiver is delivered the datagram by the operating system at the remote machine. However, due to the difficulty of synchronizing clocks in different machines and the lack of support for precise timing within some current operating systems, we cannot directly measure the network delay of packets. Instead we measure the roundtrip time in the network as seen by the network monitor. These delays can be used to extrapolate the one-way network delays.

In our experiments a source program at the traffic-generating workstation, using the standard UDP/IP protocol stack, sends a stream of datagrams to a process on a remote machine. The size and interpacket times for the datagram stream are taken from the traffic profile for voice obtained using the IVS package. Each UDP datagram fits easily into a single IP packet, and each datagram carries a sequence number incremented by the source program. At the remote machine a user-level program receives the datagrams and immediately transmits a datagram of the same size and with the same sequence number back to the traffic-generating workstation. The roundtrip time is determined by packet timestamps from the network monitor and represents endsystem processing at the remote machine as well as the time to cross the network twice.

Table 5.1 shows the three different paths through the network that this study examines. Routers are identified by the names used in Figure 5.1, with the exception of the router

[1] We used Version 2.1 of the IVS software. Future versions of the IVS software will have a more sophisticated voice protocol that does incorporate talkspurt marking [60].

Path	Routers (see Figure 5.1)	Destination Name	Destination Type
1	Olsson, Thornton	weyl.math.Virginia.edu	RS 6000
2	Olsson, Gilmer-1, Adcom	apollo.itc.Virginia.edu	Sun4 IPC
3	Olsson, Gilmer-1, VL, Hospital	crcsun1.med.Virginia.edu	Sun4 IPC

Table 5.1: Routes for Voice Transmissions.

Adcom. This router is not pictured in Figure 5.1, but in the current network it is attached to the large FDDI ring (subnetwork 226) in the middle of Figure 5.1. Path 1 consists of 3 network segments, two Ethernets with each connected to an FDDI ring. Path 2 includes 4 network segments— the source Ethernet, an FDDI ring (subnetwork 222), a second FDDI ring (subnetwork 226), and the destination Ethernet. Path 3 spans 5 network segments with 3 FDDI backbones between the source Ethernet and the destination Ethernet.

The data shown here was taken in consecutive experiments, all completed between 10:30 AM and 11:30 AM on the same weekday. (Other data sets, omitted for brevity here, corroborate the essential aspects of the measurements presented.) There was a background load on the local Ethernet segment (subnetwork 60) of 5 Mbits/s during each experiment. The network loads at remote subnetworks at the time of the experiment are not known, though the statistics in Figure 5.1 give an indication of typical workday loads. The source machine is a Sun SPARCstation IPC with a single user. The remote machines are desktop workstations that were lightly loaded while measurements were being taken.

The measured roundtrip delays for Path 1 are shown in Figure 5.2, with the negative spikes indicating dropped packets. For the first 90 seconds the network has low delay, i.e., most of the roundtrip times are in the range of 2-8 ms. However, at around 90 seconds into the experiment, the network experiences large fluctuations in roundtrip times with peaks up to nearly 200 ms. This behavior continues for approximately 60 seconds, at which point the network roundtrip times return to a lower range with a few spikes in the 30-50 ms range.

Thus, during quiescent periods, we observe low delays that indicate a very fast path

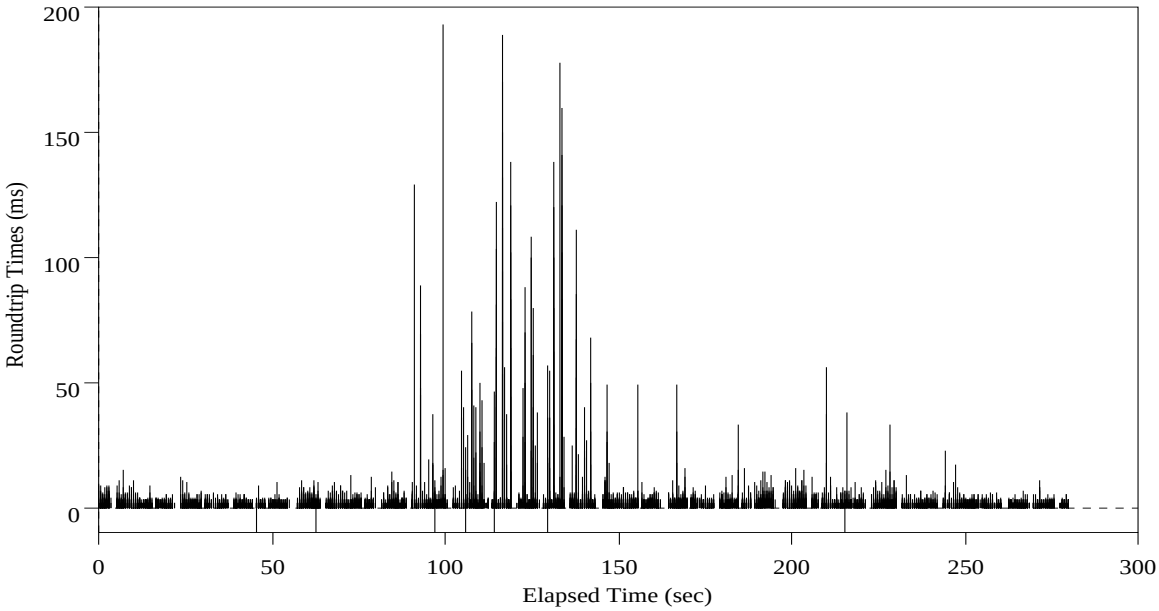

Figure 5.2: Roundtrip Times over Path 1.

across the network and through the kernel-level processing at the endsystem. Figure 5.3 gives a closer look at the delay spikes during a period of increased network delays over Path 1. In this figure the roundtrip delay of individual packets are plotted, and the large delay spikes clearly display a pattern whereby the end-to-end delays for consecutive packets associated with a spike are regularly spaced. Since the packets were transmitted from the source 20 ms apart, the 20 ms spacing in Figure 5.3 indicates that the packets were queued up at some source and then released in a burst, arriving at the network monitor back-to-back. We believe it is unlikely that processing interrupts at the endsystem would consistently produce this effect, though, since we are interested in end-to-end network delays, for our purposes it makes little difference whether these effects are caused by the endsystems or routers in the network.

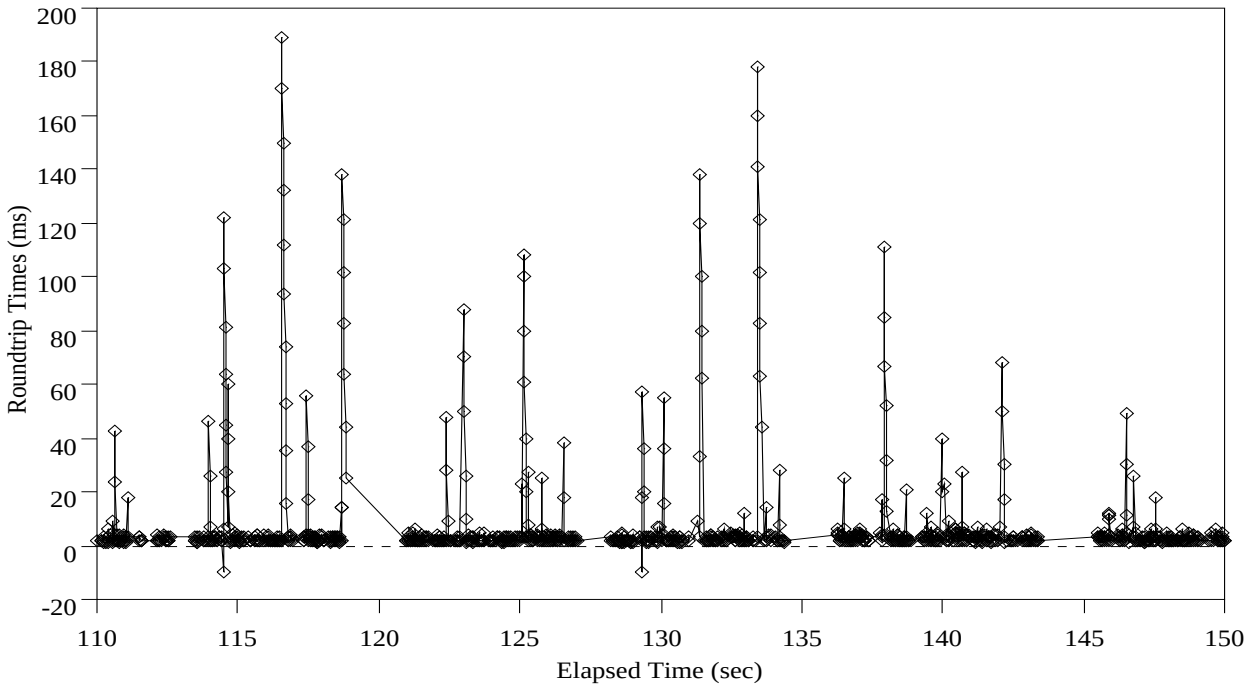

Figure 5.3: High Delays in Path 1.

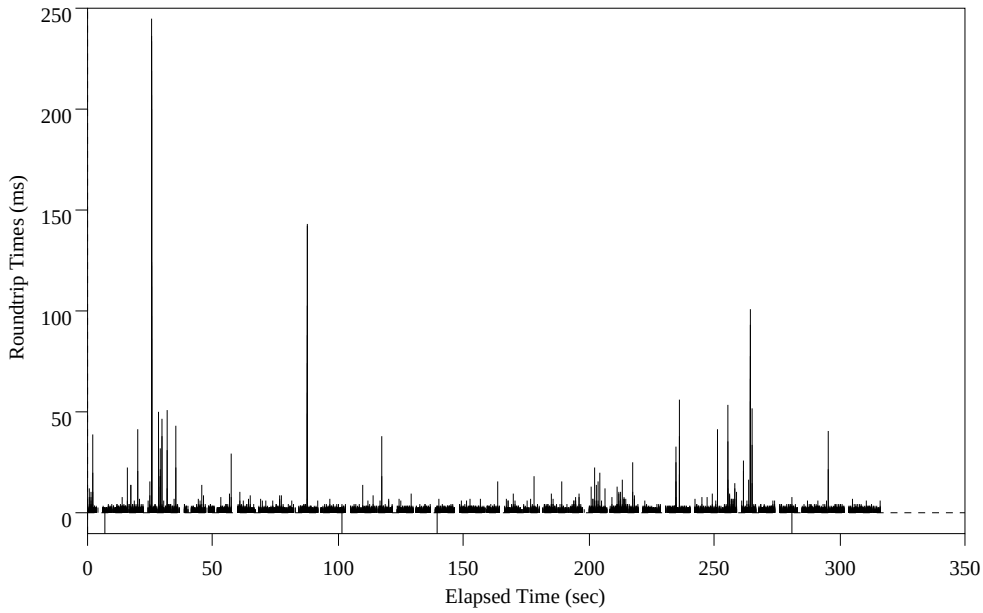

Figure 5.4: Roundtrip Times over Path 2.

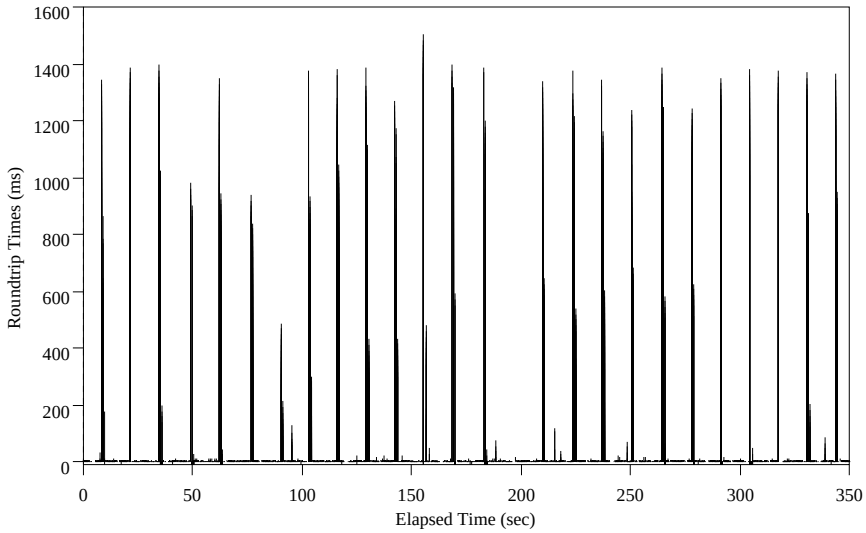

Figure 5.5: Roundtrip Times over Path 3.

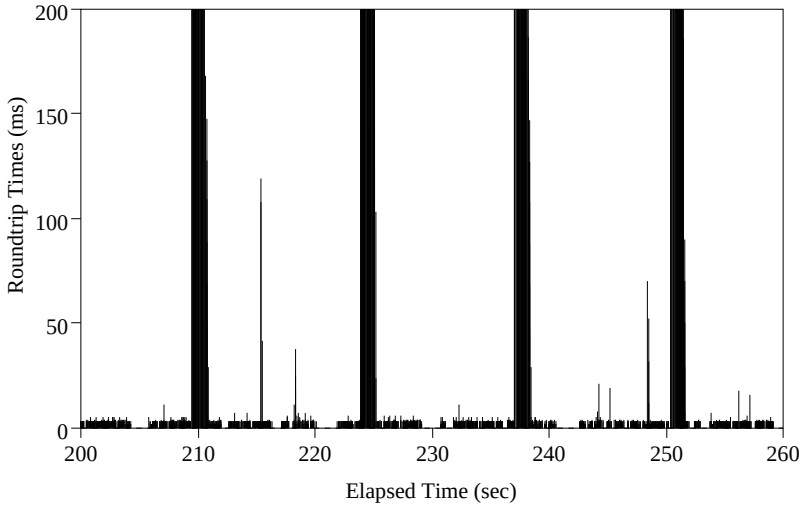

Figure 5.6: Roundtrip Times over Path 3 (Reduced Scale).

The second path crosses the most heavily loaded router in the network, *Gilmer-1*, which interconnects two FDDI backbones. The destination machine is a Sun SPARCstation IPC. The roundtrip delays, shown in Figure 5.4, are generally in the 2-8 ms range. We believe that the isolated incidents of high delay are due to periodic routing tables updates since additional experiments with this router show periodic performance degradations every 30 seconds. Figure 5.4 supports this conclusion. Close observation reveals delay spikes (of varying size) occurring at 27, 57, 87, and 117 seconds into the experiment. Based on other experiments, we conjecture that the size of the spikes is a function of network load at the router.

Path 3 was chosen because it is one of the longest possible paths in the network, crossing two Ethernet segments and three FDDI rings. The destination machine here is a Sun SPARCstation IPC with very light loads. The roundtrip times are shown in Figure 5.5, and the path turns out to have delay spikes of 1.4 seconds that appear regularly every 13 seconds. This behavior is not transient, i.e., specific to the day and time at which the experimental data shown was collected. For several days after the experiment we performed sporadic checking using the Unix *ping* utility to confirm that the observed behavior was not specific to the time of our experiment.

Figure 5.6 shows the measured delays over Path 3 on a timescale similar to that for Paths 1 and 2. The network is seen to be quite well-behaved between the large periodic delay spikes. We conclude that the data from Path 3 gives the clearest example of router-based delay problems.

5.3 Analysis of Empirical Results

We next discuss the implications of the empirical results for our analytical model and for the potential of this campus-wide network to support continuous media traffic. We first discuss the characteristics of the packet voice stream and then consider the network delays observed.

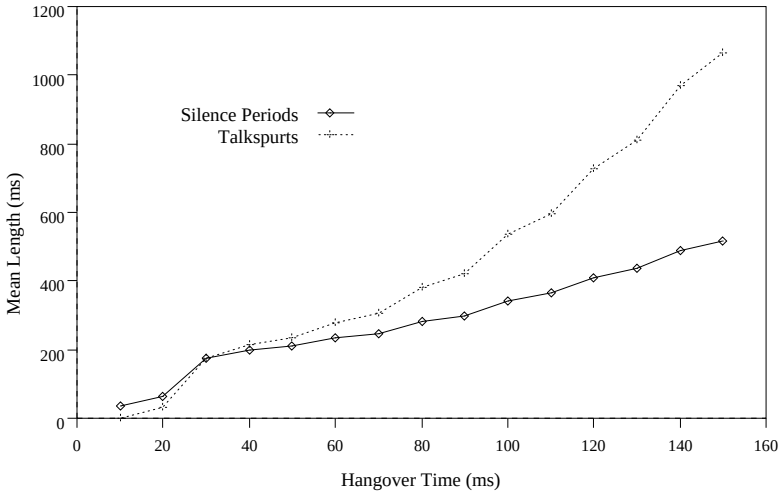

Figure 5.7: Talkspurt Size as a Function of Hangover Time.

5.3.1 Voice Traffic Characteristics

Determining talkspurt boundaries is an important protocol mechanism since it affects playback timing at the packet voice receiver. In our experiment the encoding scheme uses a silence detection algorithm, which occasionally suppresses transmission of a small number of packets during a speech activity period. If the packet voice receiver incorrectly identifies these small silence periods as talkspurt boundaries, the result will be inappropriate pauses in speech playback. Even if short silence periods represent natural breaks in speech, variations in short silence periods due to control times are more noticeable than in long periods.

When using silence suppression, the voice protocol must therefore enforce a minimum intertalkspurt time, or *hangover time*, when marking talkspurt boundaries. Figure 5.7 shows the effect of different hangover times on the average duration of talkspurts and the average duration of silence periods in our voice data. As seen in Figure 5.7 the use of even a small hangover time significantly increases the size of talkspurts and silence periods. If the hangover time is 20 ms, for example, then each activation of the silence detection algorithm effectively creates a new talkspurt, resulting in silence periods of 35 ms on average. A hangover time of 30 ms joins speech activity periods separated by a single packetization

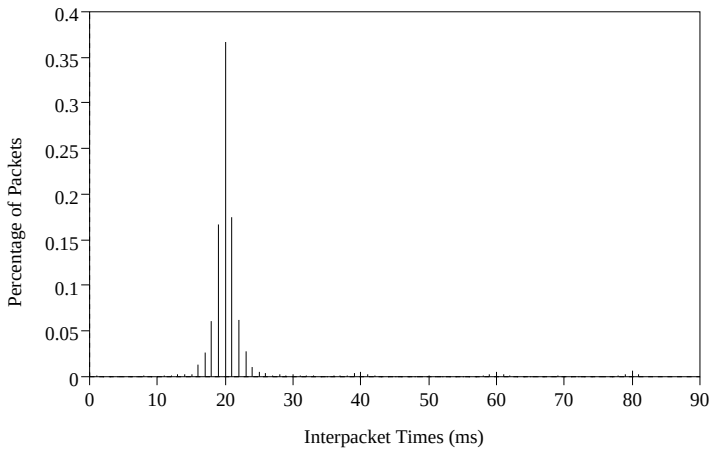

Figure 5.8: Density Function of Interpacket Times at the Source.

interval, and the average duration of silence periods increases to around 200 ms. Subsequent increases in the hangover time cause a more gradual increase in the duration of talkspurts and silence periods. We conclude that, to achieve silence periods of relatively long durations (i.e., 250-300 ms), a hangover time of approximately 100 ms is necessary, in line with [24].

Figure 5.8 shows the frequency distribution for the interpacket times of packets in the voice stream leaving the traffic-generating workstation. In line with the analytical model, the figure shows that the packetization interval generally dominates protocol processing at the transmitter. Approximately 83% of all packets are transmitted on the network with an interpacket time in the range of 18-22 ms and over 91% with an interpacket time in the range of 15-25 ms. (Recall that timestamping in the experiments is accurate to within ± 1 ms.) Approximately 1% of packets have interpacket times between 39-41 s, 1% between 59-61 ms, and 1% between 79-81 ms. Packets in these intervals are being transmitted after one, two, or three consecutive packets have been suppressed by the silence detection algorithm. The set of packets that consists of the first packet in each talkspurt, which accounts for approximately 5% of all packets, is excluded from this frequency distribution.

The frequency distribution for the number of packets in a talkspurt is given in Figure 5.9. the empirical data demonstrates a high variance in the number of packets, with many talkspurts containing less than ten packets. This behavior is corroborated in [25].

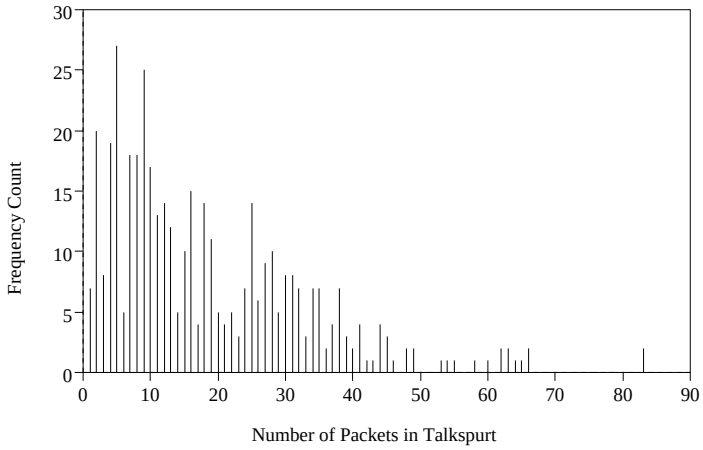

Figure 5.9: Packets in a Talkspurt.

This distribution improves the likelihood of continuous playback for a talkspurt since the probability of a jitter gap is reduced by having fewer packets in the talkspurt. Thus, given this empirical frequency distribution, the retransmission probabilities from our model, when based on a single output derived from the mean number of packets in a talkspurt (as was the approach in the numerical examples in Chapter 4), will underestimate the true probability of successful recovery from packet loss.

5.3.2 Network Measurements

5.3.2.1 Observations

The first observation about the measured network delays is that all three paths exhibit delay behavior feasible for the transport of real-time packet streams, though the very large periodic delays on Path 3 seem likely to degrade the transmission quality. Excepting the extreme delay spikes on Path 3, the measurements reveal a network that can be roughly characterized as changing between a quiescent state with low stable delays to an unstable state during which delays and delay variations are generally high. For instance, Path 1 demonstrates an unstable period for approximately 60 seconds, and Path 2 exhibits at least two periods of moderately high delays that last for approximately 20 seconds each, i.e., in Figure 5.4, at around 30 seconds of elapsed time and at around 260 seconds of elapsed

time. Our data suggests that at least some increased delays are related to periodic router behavior, which would correlate with observations in [17, 44], though this periodic behavior does not explain all the high delays observed.

Packet losses are rare, but they do occur in the experiments. All losses that were observed are isolated single-packet losses. Losses represent a very small percentage of the total packets transmitted, e.g., in the data for Path 1, for example, there are 7 packet drops out of the 7700 packets observed. Since our quality measure is the continuity of playback for a talkspurt, however, a more meaningful measure is the percentage of talkspurts that experienced at least one packet drop. Packet losses occur in 1.7% of the talkspurts in the Path 1 experiment. If we isolate attention to the 60-second high-delay period for Path 1, a packet loss occurs in 4.5% of talkspurts. Many of the losses that appear in the empirical data occur during periods of low delay in the network, suggesting that these losses are not strongly correlated with high network delays, as found in losses for wide-area transmissions across the Internet [44].

5.3.2.2 Empirical Probabilities

For comparison with our theoretical results in Chapter 4, here we calculate the probability of continuous playback of a talkspurt using the empirical delay measurements. Our delay measurements are for roundtrip times in the network. However, in the following analysis, these delays will be treated as conservative estimates of one-way network delay. We choose this approach since the behavior shown in Figure 5.3 gives evidence that packets with high delay accumulate their delay at a single point in the network, e.g., at a congested router. Thus, if we were to employ the commonly-used strategy of dividing roundtrip delays by half in order to estimate end-to-end delays, we risk severe underestimation of the largest network delays. Also, by using roundtrip times, we obtain conservative measurements on retransmission behavior.

In Figure 5.10 the *0-error* probabilities for the delay measurements over all three paths have been empirically determined and plotted over variations in the control time. By "empirically determined" we refer to the reconstruction of the playback schedule using

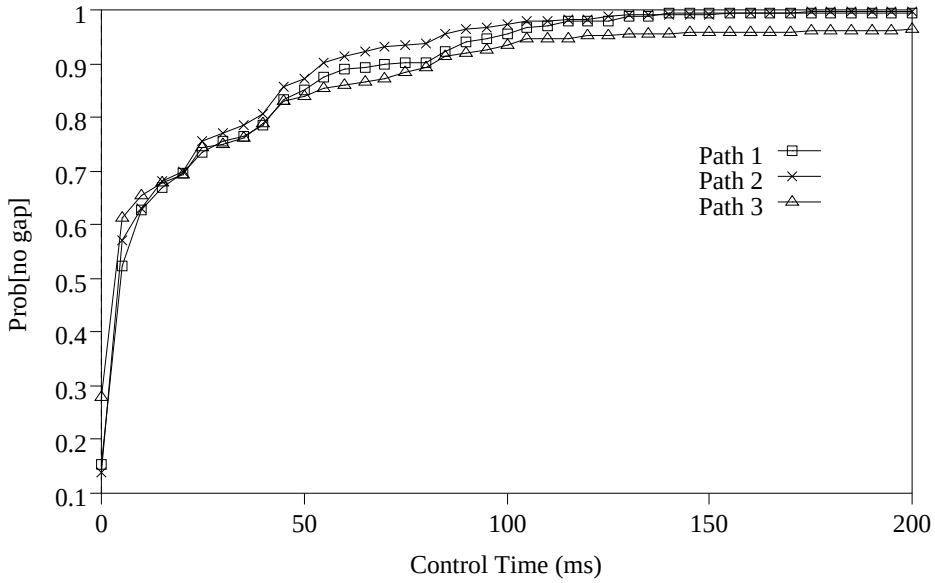

Figure 5.10: Control Times and the Elimination of Jitter Gaps.

measured network delays and a given control time value. For our purposes we assume a fixed control time value for all talkspurts. We observe that a fixed control time of approximately 120 ms will achieve continuous playback at the receiver for at least 98% of the talkpurts on Paths 1 and 2, though only 95% on Path 3. For Path 3 a control time of 200 ms achieves continuous playback for only 96% of the talkspurts, that is, the extremely high delay spikes cannot be compensated for by any reasonable size control time at the receiver.

Figure 5.11 gives the *k-error* curves for the data from Path 1. To calculate the curves in scenarios with errors, an estimate of the roundtrip time for retransmissions is required. We estimate this delay by taking twice the average of the network delay of the four packets arriving after the packet presumed lost, i.e., we look into the future to estimate the network delay that the retransmission will encounter. As in the analytical model, the *k-error* curves are calculated under the assumption that a single burst loss occurs with uniform distribution within a talkspurt.

Note that the non-intuitive inversion for small control times in Figure 5.11, i.e., three-error loss is more likely to be recovered than a one- or two-error loss, is purely an artifact of our calculations. This anomaly occurs due to the fact that there are a number of short

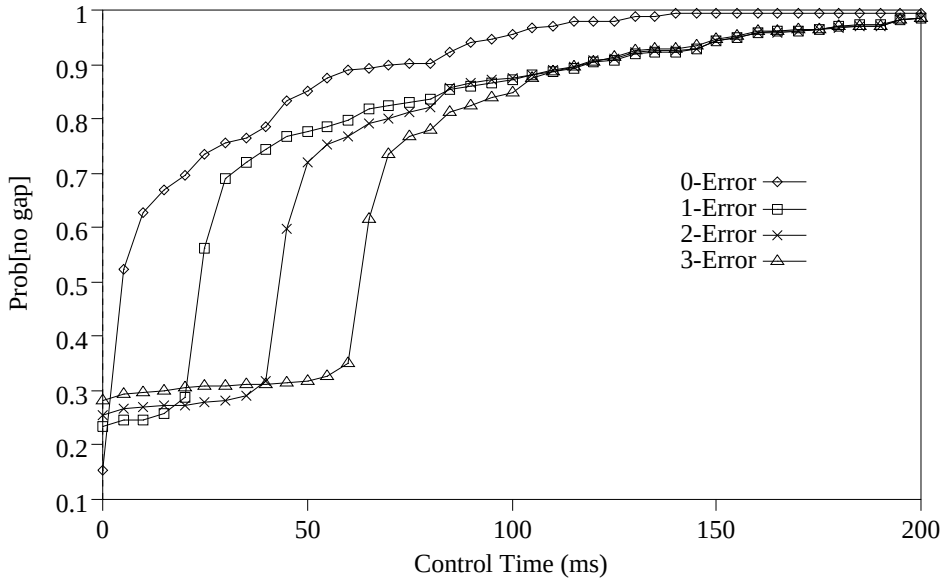

Figure 5.11: Control Times and Retransmission Probabilities for Path 1.

talkspurts in the data and each error scenario considers a different set of loss positions. When the probability of loss recovery is very low, the consideration of fewer loss positions favors the scenarios with larger burst losses.

More significantly, from Figure 5.11 we observe that, in the range of control time values needed for jitter control, the probability of successful retransmission for moderate-size burst losses is high. The empirical data thus supports the conclusion that retransmission is a feasible strategy that provides a high degree of error coverage.

The most important result evident in Figure 5.11 is the reproduction of the characteristic "S"-shaped curve from the analytical and simulation studies. That is, we have experimentally verified the curves for retransmission success over variations in the control time predicted in our models. The "knee" of the curve for the probability of successful recovery of k lost packets is located at the control time value of k times the packetization interval. In this particular network the curves rise rapidly after the "knee", reflecting the large amount of traffic that experiences low delays, followed by a long tail, due to the difficulty of recovering losses if they occur in the period of high delay and high delay variations. Empirical retransmission probabilities calculated with the network measurements on Paths

2 and 3 exhibit curves very similar to those in Figure 5.11.

5.4 Summary

Our goals in this chapter were first to consider the impact of empirical measurements on our analytical retransmission model, and second to assess the feasibility of campus-wide networks supporting near-term deployment of continuous media applications. We summarize the conclusions of the study below.

Measurements of voice traffic confirm the assumption in the analytical model of a deterministic traffic pattern for voice. Little variation in interpacket times is observed at the traffic-generating workstation, indicating that protocol processing at the transmitter is dominated by the packetization interval. The number of packets in each talkspurt exhibits high variance in our voice sample. In Chapter 4 it was shown that the retransmission probabilities calculated from the analytical model, which assumes a fixed-length talkspurt, are not strongly sensitive to the length of a talkspurt. However, due to the high variance for the length of actual talkspurts, multiple runs of the model with different settings for the talkspurt length may be needed in order to capture the nature of real voice traffic.

The network delays measured in this study are not well-suited to being modeled with a single continuous delay distribution. The measurements clearly show that the network delay distribution shifts with time, implying that more than one distribution is needed over long time periods. From the measurements for this network, the network delay behavior appears to separate into a low-variation distribution for periods of low delay and a high-variation distribution for the periods of instability.

Retransmission probabilities generated with the empirical data, however, support the conclusion from the simulation and analytic studies that retransmission is feasible under the control times required to fully compensate for delay jitter in the network. In addition, these empirical probabilities show that the overall effect of delays on retransmission behavior follows the basic pattern predicted by the model, i.e., the characteristic "S"-shaped curves from the models are empirically verified.

Packet losses did occur during network transfers. The amount of loss was quite small over the entire data set, but the impact on quality as measured in the playback continuity of talkspurts appears to be non-negligible over some time intervals, specifically over the 1-minute period of high delay observed in Path 1.

Based on the measurement study it can be concluded that the University of Virginia campus-wide network has good potential for supporting real-time traffic streams. Sporadic high delays in the network, however, are a threat to quality. We speculate that the high delays observed in the measurements are largely an artifact of router design, and that, as the importance of continuous media traffic increases, mechanisms will be developed to reduce or eliminate this behavior. But the sensitivity of continuous media traffic will make the transition to supporting real-time traffic in the network a difficult one. The extreme delays observed on Path 3 in our measurements stand as an example of anomalous network behavior that is of little consequence to data traffic and hence goes unnoticed, but which is disastrous for real-time traffic.

Chapter 6

Conclusions

As the trend towards accommodating audio and video in digital processing environments accelerates, computer communications must meet the challenge of providing services for delay-sensitive traffic streams. At first these services will emphasize connectivity and relatively little emphasis will be placed on refinement of the network delivery mechanisms. As integrated services networks mature and traffic demands increase, service providers will increasingly focus on the construction of efficient services that fully exploit statistical multiplexing gains in the network. Delay-constrained error control, in which endsystems recover packet losses due to temporary network congestion, will allow aggressive use of network resources without a catastrophic loss in transmission quality.

This thesis contributes to the construction of such services by providing a comprehensive and fundamental study of retransmission-based error control for the distribution of digital continuous media over packet-switched networks. We have formulated a novel retransmission scheme that avoids significant drawbacks associated with proposed open-loop error control techniques for continuous media. A major barrier to the consideration of retransmission-based error control in the past has been the lack of a methodology by which to assess its feasibility and effectiveness in the presence of stochastic network delays. We have developed analytic techniques that provide a quantitative measure of retransmission performance, based on the quality experienced by the continuous media stream. We performed simulation and analytical studies of realistic transmission scenarios and supported this work with empirical measurements of actual voice transmissions. Our results lead us to

conclude that, contrary to the current wisdom, retransmission-based error control can provide significant error coverage for continuous media communications in many packet-based transmission scenarios while respecting delay constraints that are critical to the overall distribution quality.

We now highlight the contributions of this thesis by chapter and identify directions for further research.

6.1 Summary of Contributions

In Chapter 1, we place the problem of error control for continuous media communications in the context of maintaining the overall quality of the transmission, which implies respecting the delay constraints of the stream. Error control in conventional reliable end-to-end protocols are delay-insensitive and therefore inappropriate for continuous media.

In Chapter 2, we survey the techniques proposed for error control with continuous media streams. While the literature on error control is quite large, error control under real-time constraints is a relatively new topic. Retransmission schemes have generally been eschewed or dismissed altogether. We conclude that, if timely retransmissions are achievable, retransmission is attractive since it imposes little overhead on network resources and since alternative techniques have notable drawbacks with respect to complexity, portability, and cost.

In Chapter 3, we define a novel retransmission-based approach to delay-constrained error control, the Slack ARQ scheme. Feasibility of the approach for LAN-based packet voice distribution is explored through a simulation study. A unique component of the simulation model is the development of a performance metric for retransmission effectiveness that is based on the overall transmission quality experienced by the delay-sensitive packet stream. Experiments with the model presented in this chapter show that the buffering times required for Slack ARQ are on a timescale that allows significant error coverage through retransmissions while respecting the delay requirements of interactive voice.

In Chapter 4, we develop an analytical end-to-end model for Slack ARQ. The model is

quite general, abstracting away the details of encoding schemes and details of the underlying network architecture. Using the model we develop analytical expressions for the performance metric developed in the simulation model. The analysis provides, without resorting to lengthy simulations, quantification of parameter interactions, such as the packetization interval in the protocol and the network delay distribution, that affect retransmission success. This methodology is an important contribution. No previous studies have developed techniques by which the effectiveness of retransmission for a continuous media stream could be evaluated over variations in network and protocol parameters.

In Chapter 5, we report the results of an empirical study on the traffic characteristics and network delays of actual packet voice streams transmitted across a contemporary high-performance campus-wide network. Our study is of particular interest since large multiple-segment LANs are likely candidates for near-term deployment of continuous media applications and since there is little previous empirical work on continuous media streams in this environment. We conclude that in general current large LANs can support continuous media, though our measurements indicate that occasional periods of high delay will threaten the quality of these real-time traffic streams.

The empirical measurements support crucial assumptions underlying the analytical model for retransmission in the following ways. First, we observe relatively small delay variations for packets at the transmitting endsystem, leading to a traffic profile, as in the model, dominated by the packetization interval. Second, retransmission probabilities calculated using the empirical network delay measurements concur with the theoretical findings in that the control time required at the packet voice receiver to compensate fully for delay jitter in the network also provides for significant error coverage under the Slack ARQ scheme. Third, the curves representing the empirically determined retransmission probabilities have the characteristic shape of the curves generated from our simulation and analytical models. That is, at least for one real network, our approach models the retransmission probabilities in an accurate fashion.

In Appendix A, we describe the design, implementation, and evaluation of a novel retransmission-based error control scheme embedded in XTP, a next-generation transport

protocol. Constructed as a modification to an existing unreliable service defined by XTP, our service allows delay-sensitive clients such as continuous media applications to express their error tolerances to the underlying XTP error control algorithm in order to tune the aggressiveness of the retransmission algorithm in recovering lost packets. By exploiting functionality implemented for reliable modes of XTP, particularly management of retransmission buffers, the new error control service was created with very little additional protocol code or complexity. Experimental network transfers confirm that the new service can provide limited error coverage while maintaining essentially the same delay characteristics as the unreliable stream communication in XTP.

6.2 Future Work

Our work has many possible future directions.

One area is the development of implementation techniques to take advantage of our analytical results. This extension of the thesis has a number of challenges that are likely to provide further insight into the applicability and power of delay-constrained error control. Two important issues include development of an on-line approximation of our analytical approach and adapting to changes in the network delay distribution during transmissions.

Hybrid approaches to error control represent another interesting area. Just as with hybrid FEC/ARQ for data communications, we expect our Slack ARQ scheme to be used in conjunction with other techniques, when available. In some ATM environments, for example, cell-level forward error correction hardware will be available for use in conjunction with a Slack ARQ scheme in the higher layer protocol. Interesting trade-offs arise in determining the parameters for the two error control schemes, e.g., packet size, degree of overcoding, and amount of cooperation between the two schemes, that strikes the best balance between performance and network overhead.

Our investigation in this thesis has been largely conducted in the context of interactive packet voice, which, as we point out in Chapter 1, can be viewed as a very simple model of

video traffic. The extent to which more sophisticated models for video are needed is currently unknown, but in any case a more thorough investigation of video traffic is warranted. In addition to traffic modeling, many video transmission scenarios will require consideration of the buffer requirements for retransmission. An interesting approach for on-demand video, for example, is to use feedback at the receiver to control the rate of delivery from the transmitter. From an error control perspective, the buffer at the receiver should be maintained such that, without overflowing and losing data, it provides enough buffer time for retransmissions. The analytical end-to-end model developed in this thesis represents a good framework for studying the closed-loop algorithms for both rate control and error control in this problem and, by extension, similar rate control problems for packet video.

Extension of our concepts to multiple parallel streams is another possibility. In multimedia applications, related parallel streams often have synchronization requirements that will interact with other quality issues such as error control. Streams with higher priority or greater sensitivity to loss will impose resynchronization actions on less important streams, requiring a more complex structure for evaluating the trade-off between error recovery and stream delay constraints.

Error control in multicast communications has become increasingly important and especially so for continuous media. A number of applications for continuous media, e.g., conferencing applications, have a natural need for multicast distribution. In multicast transmission scenarios, the loss of packets at a single receiver may in general affect the progress of all other receivers, and hence the quality of the transmission for the multicast group, and not just the individual receiver, should determine protocol control actions. In Appendix A we note the value of our work there to the multicast error control defined in XTP, and schemes for other protocols are an exciting research area.

Appendix A

A Lightweight Limited-Retransmission Service for the Xpress Transfer Protocol

A.1 Introduction

We define a new class of transport layer service for applications that desire to trade off data completeness for latency considerations. Referred to as a Partially Error-Controlled Connection (PECC), the new service is developed as an enhancement to a next-generation transport protocol, the Xpress Transfer Protocol (XTP) [57]. PECC adapts the retransmission algorithm within XTP to construct connection-oriented communications under which the retransmission of lost packets occurs only when it will not add additional delay to the data delivery. The PECC client provides the transport layer protocol with application-specific values that allow for estimation of the time available for retransmission and give explicit knowledge of the error tolerance of the client.

Figure A.1 presents a perspective on end-to-end communication paradigms that focuses on data sequencing and data completeness. When an application requires fully in-order sequencing and no data loss, the transport layer service provides a reliable connection. Without the aspect of sequencing this service becomes an acknowledged datagram, and if no efforts are made to ensure delivery, the service degenerates to datagram. XTP fills the fourth option, offering a novel *no-error mode* connection in which the functionality of an XTP connection, e.g., in-order data delivery, is provided but with no active error control.

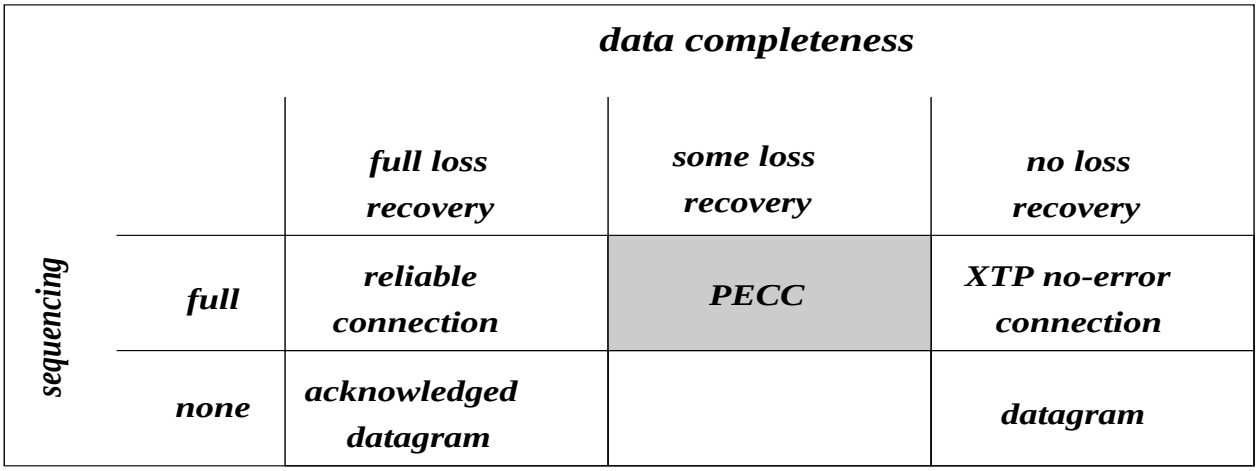

Figure A.1: Sequencing and Error Control in Communication Services.

Our new service fills a middle position between the reliable connection and the XTP *no-error mode* connection by providing in-order data delivery with limited loss recovery. While this type of service is not available in conventional service models, it is useful for emerging delay-sensitive applications such as continuous media.

The rest of this appendix is organized as follows. In Section A.2 we describe the service interface and in Section A.3 the modifications to the XTP retransmission algorithm that implement a PECC channel. In Section A.4 we present the implementation strategy used to graft the PECC service onto XTP with very little additional complexity. We report on measurements of our implementation under different parameter settings in Section A.5, and we summarize our conclusions in the closing section.

A.2 PECC Service Interface

The PECC service interface has four parameters: *fifo_min*, *window_length*, *window_density*, and *max_gap*. Under the PECC communication model, the XTP receiver logically places the data received from the network into its buffers, which are first-in, first-out (FIFO) structures emptied by the PECC client. The parameter *fifo_min* indicates the minimum amount of data in bytes that must be queued for the client before the XTP receiver will request a

retransmission of lost data. If the client empties its FIFO at a fixed or nearly fixed rate, as with continuous media applications, the depth of the FIFO of the client translates directly into an amount of time before the client will consume the data currently in its FIFO. The *fifo_min* value then represents the PECC client's estimate of the minimum time required for lost packets to be recovered by retransmission from the source without the client's FIFO underflowing.

In the PECC algorithm, when retransmission cannot take place in a timely fashion, data is "skipped". That is, in order to make progress, the PECC receiver marks buffers that have no client data in them as having been correctly filled and updates its internal state accordingly. When the client is delivered these buffers, the client is receiving "dummy" data.

The parameters *window_length* and *window_density* describe the tolerance of the client for the frequency and duration of errors during the transfer. The underlying protocol implements a sliding data window of length *window_length*. The service guarantee to the client is that, if latency considerations result in more than *window_density* bytes of missing data having to be skipped in any interval of *window_length* data bytes, the PECC service will notify the client of the service violation. Finally, *max_gap* represents the maximum number of missing bytes that will be skipped during any one pass through the PECC algorithm. *Max_gap* controls the rate, relative to new data packets arriving, at which data is skipped. Note that using packet arrivals to drive the aggressiveness of the retransmission algorithm is motivated by the fact that this rate characterizes the amount of fresh data available to the receiving PECC client. The PECC client can control this rate with the window parameters, i.e., *window_length* and *window_density*.

The PECC service reports failure when the service guarantee on the spacing and size of missing data must be violated in order for the data transfer to continue in a timely fashion. When a failure occurs, the PECC implementation provides the client with an indication of the service failure and continues the data transfer.

A.3 Retransmission Algorithm

We now describe in detail the retransmission algorithm embedded in the XTP receiver that implements the PECC service. Presentation of the algorithm requires the terminology of buffer management at the XTP receiver, which we present in Figure A.2. The figure shows buffers at an XTP receiver. Dark areas represent data received, while light areas represent portions of the data stream not yet received at the XTP receiver. In the situation depicted, data has arrived out of order, and hence there are gaps in the buffered data. Since XTP supports selective retransmission, the XTP receiver supports buffering for packets that arrive out of order. Sequence numbers, which are byte-based in XTP, increase from left to right in the buffers in the figure, e.g., the packet shown arriving from the network, which is assumed to represent new, in-sequence data, will be placed into the buffers at the right as shown.

Figure A.2 shows three state variables associated with the XTP receiver's buffers: *dseq*, *rseq*, and *hseq*. The value of *dseq* is one greater than the highest sequence number of data delivered to the client; the value of *rseq* is one greater than the highest sequence number of data received in sequence; the value of *hseq* is one greater than the highest sequence number of data received. Note, once the packet shown arriving from the network is placed in the buffers, *hseq* will be increased by the size of the payload of the packet. The term *current gap* in the PECC algorithm refers to, upon arrival of a packet from the network, the data between *rseq* and the next data buffered at the receiver. If no data is present beyond *rseq* when a packet arrives out of order, the current gap is the data between *rseq* and the sequence number of the first byte in the arriving packet.

For the XTP receiver, skipping data means increasing the value of *rseq*, even though the client data for this part of the sequence space is not available. When data is skipped, the XTP receiver advances its state as though the skipped data were correctly received, and the new state of the receiver is eventually reported through normal protocol procedures to the XTP transmitter. In this way the data transfer makes progress, and data sequencing is handled as in a reliable connection.

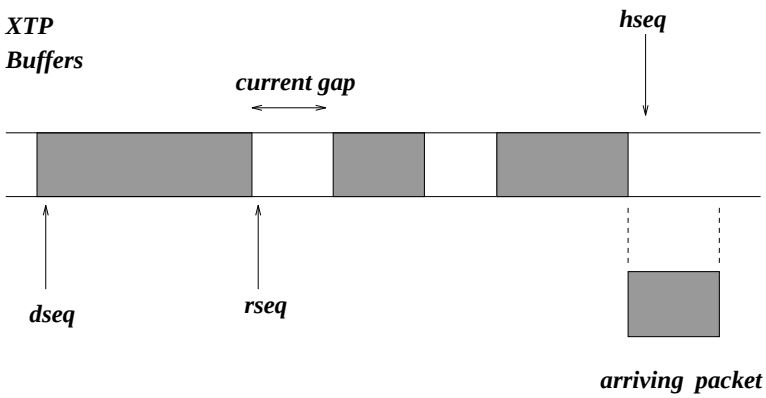

Figure A.2: State Variables at the XTP Receiver.

We will now discuss the PECC algorithm itself, which is shown in pseudo-code in Figure A.3. The algorithm is invoked by the arrival of an out-of-order packet, and it begins by initializing two variables, *curr_gap* and *win_credit*. The value of *curr_gap* is the size of the current gap as defined above. The value of *win_credit* is the maximum number of bytes that can be skipped in the current window, as determined by the value of *window_length* and *window_density* in the current PECC configuration and the amount of data already skipped data in the current window.

The first decision point (line 1) compares the current amount of data available to the client but as yet unread with the the value of the parameter *fifo_min*. If the amount of unread data exceeds *fifo_min*, the PECC receiver assumes that a retransmission can occur in a timely fashion and sends a control packet to the transmitter requesting a retransmission of all missing data in its buffers (line 2). Note that the packet processing to issue a control packet and request retransmissions is the same as in a reliable transfer; no new protocol code need be written.

If the condition in line 1 does not hold, line 4 then determines if the current gap in the receiver's buffers is allowed to be skipped. The test in line 4 involves comparing the size of the current gap to the minimum of the value of *max_gap* and *win_credit*. If the

Variable Definitions:
dseq—one greater than the highest sequence number of data delivered to the client.
rseq—one greater than the highest sequence number of data received in sequence.
pseq—sequence number of first byte of data in arriving packet.
Invoking Event:
Packet arrives at the PECC receiver with $pseq \neq rseq$.

```
        begin
0.          initialize curr_gap and win_credit;
1.          if ( rseq − dseq ≥ fifo_min )
2.              send control packet for retransmission;
3.          else
4.              if ( curr_gap < min{max_gap, win_credit} )
5.                  skip curr_gap bytes;
6.              else
7.                  skip min{max_gap, win_credit} bytes;
8.                  if ( rseq − dseq ≥ fifo_min )
9.                      send control packet for retransmission;
10.                 else
11.                     skip curr_gap − min{max_gap, win_credit} bytes;
12.                     record service violation;
        end
```

Figure A.3: Retransmission Algorithm at the PECC Receiver.

size of the current gap is sufficiently small, then the gap is skipped, and packet processing continues under the new state. If the current gap in the data cannot be skipped, line 7 of the algorithm specifies skipping the maximum acceptable amount of data as determined by the test in line 4.

The test from line 1 is now repeated in line 8 to see if, based on the new *rseq* value obtained after skipping bytes in line 7, there is now sufficient buffer time for the current gap to be retransmitted. If the test in line 8 is successful, packet processing continues as it would in a reliable transfer. If the test in line 8 fails, there is not enough time to recover the

missing data and the entire current gap is skipped. At this point the PECC receiver records the fact that the PECC service guarantee on the size and frequency of gaps in the data stream has been violated. The data transfer continues, and the PECC client is signaled in an asynchronous manner of the service violation.

Our hypothesis is that a PECC service can provide an intermediate service between a fully reliable connection and the XTP *no-error mode* connection. The first observation is that the PECC service can mimic each of these bounding services. If the *fifo_min* variable is set to 0, then there is always time enough to recover lost data since there is always at least 0 bytes in the client's FIFO (line 1). For *no-error mode* service, the *fifo_min* and *max_gap* variables are given very large values and no window criteria is specified, e.g., *window_length* and *window_density* are set equal. In this case the PECC algorithm always decides there is not enough time to retransmit (the test in line 1 fails) and then always skips the missing data in full (the test in line 4 and line 8 fails).

A.4 Lightweight Implementation Strategy

The sender-driven architecture of the XTP protocol offers a convenient framework for a low-overhead implementation of the PECC service. Specifically, the XTP receiver uses the technique of skipping data to maintain the desired rate of progress. The XTP sender is unaware of the more sophisticated error-reporting algorithm, and there is no PECC-specific code on the sending side.

We use this strategy of modified error-reporting algorithm to implement the PECC service as a lightweight enhancement to XTP. It is lightweight in two ways. First, it does not require any change to the XTP protocol definition. The PECC mechanisms can be introduced selectively at individual XTP nodes such that full interoperability with other end-systems is preserved. This transparency is of particular interest in the case of the multicast communication in XTP. Under the PECC scheme receivers in an XTP multicast connection can have different error tolerances using the PECC interface, and the effect places no additional processing burden on the XTP transmitter. The PECC service is also

lightweight in that it does not interfere with normal packet processing when a PECC-based transfer is error-free, nor does it add any processing burden to other protocol functions.

Our implementation of PECC is embedded in a software implementation of XTP. The modifications made to the XTP code include an interface call to invoke the PECC behavior and a small amount of additional code inside the module for processing packets received from the network. Once a receiving XTP context is opened, the client invokes the PECC option with an interface call on the receiving side of the transfer. All other XTP interface routines are unchanged.

A.5 Experiments

The software XTP implementation in which the PECC algorithm is embedded runs as a client process on Sun-4 workstations with the User Datagram Protocol as its underlying network service provider. This version of the UVA XTP code uses the lightweight process library provided in the SunOs 4.1.1 distribution to handle the shared-memory communication between the logical layers of the UVA XTP architecture [54]. Obviously, this XTP implementation environment is most useful as a tool for studying protocol workings, which is our purpose, rather than for actual continuous media transfers. The experiments in this environment are not intended to represent performance data but to serve as a proof of concept for the PECC service and to provide insight into the workings of the algorithm, at least for a particular network environment.

The experiments involve the timed transfer of 1000 buffers of client data. Our approach is to measure the time necessary for the entire transfer of 1000 buffers, and then to extrapolate from these numbers the degree to which retransmissions stall the progress of data delivery from the network. The buffers are transmitted using a non-blocking reliable SEND primitive defined in the UVA XTP interface. The last buffer is sent with a blocking reliable SEND primitive, after which the timer is stopped. The buffers are all transmitted across a single XTP connection, which operates under reliable, *no-error*, or PECC modes as determined by the configuration of options chosen. The reliable and *no-error mode* transfers

fifo_min	*max_gap*	Total Transfer Time (sec)	Buffers Skipped	Buffers Dropped in Net
0	0	43.7 [41-49]	0	69.7 [41-100]
1000	1000	38.0 [37-40]	66.2 [44-100]	66.2 [44-100]
1	1	37.8 [37-39]	16.8 [14-20]	56.4 [39-69]
3	3	38.3 [37-40]	40.3 [29-48]	59.8 [39-76]

Table A.1: Performance of PECC Service under Various Configurations.

use only XTP-defined mechanisms. For the PECC transfer, on the sending side the XTP connection is opened for reliable transfer while on the receiving side the PECC interface call activates the new error reporting algorithm.

For the experiments artificial packet loss is introduced into the network. The transmitting side drops packets in bursts, and packets carrying data retransmissions may be dropped. The number of packets between successive error bursts is determined by a truncated exponential distribution as is the number of packets suppressed in each error burst. Burst durations are limited to nine consecutive packets in order to keep overall data loss in the experiment to a small percentage of total data transferred. The net effect is to produce packet losses in the range of 5% of the total data transferred. While unrealistically high for a stable network, this error rate ensures that the PECC code is heavily exercised.

Table A.1 shows the results of 1000-buffer transfers under various PECC configurations. The parameters *window_length* and *window_density* were rendered inactive for this experiment. For convenience the unit of measure for all parameters is given in application buffers, which were very small (4 bytes). Small buffers were used so that each application buffer was carried in a single XTP packet. [1]

The columns of Table A.1 present, from left to right, the values of PECC parameters *fifo_min* and *max_gap*, followed by the measured total time for transferring 1000 buffers,

[1] Experiments conducted with larger buffers support the conclusions drawn from the small-buffer experiments. While the total transfer times are larger, the large-buffer experiments offer no substantially new insights and are therefore omitted here.

the total number of buffers skipped at the PECC receiver, and the total number of buffers (XTP packets) dropped by the random packet-discard mechanism in the experiment. The values for network statistics are given as the average value over fifteen 1000-buffer transfers displayed beside the minimum and maximum values for any individual transfer in the set of transfers.

The first row of Table A.1 shows a PECC configuration that ensures fully reliable delivery. Since all dropped data is being recovered through retransmissions, the total number of packets sent is slightly higher in these transfers, which results in the total packets dropped being slightly higher than the other PECC configurations. As expected, the average transfer time is longer, in this case about 6 seconds, than for the other PECC configurations. The significance of this number is that it implies that the PECC client at the receiving side was blocked waiting on retransmissions at some points during the transfer.

The second row of Table A.1 shows a PECC configuration that effectively provides a *no-error mode* service. For the *no-error mode* case, the amount of data lost is exactly the number of bytes contained in the buffers that are (artificially) lost in the network—that is, the receiver skips all gaps in the data stream. *No-error mode* transfers finish more quickly than reliable transfers, and all the other PECC configurations take essentially the same amount of time as the *no-error* configuration. This is exactly the desired behavior since the PECC algorithm is designed to suppress requests for data retransmission when such a request will cause the client's FIFO to underflow. Thus, the PECC configurations only perform those data retransmissions that do not add additional delay to the data delivery service, as evidenced by comparison with the total transfer time for a *no-error mode* connection.

Configurations such as those in the first two rows of Table A.1, i.e., configurations in which *fifo_min* is less than or equal to *max_gap* and no window criteria is used, provide a thresholding service. All losses of less than *max_gap* buffers are skipped immediately. Larger gaps are partially skipped, after which the remaining part of the gap may be recovered through a retransmission. Since the PECC algorithm is invoked by every packet arriving out-of-order, these threshold services do not guarantee, however, that the largest run of

fifo_min	*max_gap*	*window length*	*window density*	Transfer Time	% of Buffers Not Recovered	Max Buffers Skipped in Window	Fails
1	3	10	3	37.8	67%	3.4	2
3	3	10	3	37.8	67%	3.9	3
3	3	30	3	37.6	76%	8.0	14

Table A.2: PECC Configurations with Window Criteria.

"dummy" data delivered to the PECC client will be of size *max_gap* bytes. As each new packet arrives, the first *max_gap* buffers in a large gap will be skipped. Nonetheless, by thresholding at 1 buffer, the PECC configuration in the third row of the table reduces buffer lost to an average of less than 1.7% of the total data transferred where a *no-error mode* connection for the same transfers would have lost, based on actual measurement of the number of dropped packets, 5.6% of the data. As seen in the fourth row of the table, thresholding at 3 contiguous buffers per skip raises the average amount of skipped data to 4% of the total data when 5.9% of the data is being dropped by the network.

In Table A.2 we performed the experiment with parameter configurations that activate the window criteria in the PECC algorithm. Here the table presents only the average values for measured quantities, gives the percentage of buffers unrecovered, and includes two window-related measures— the average of the maximum number of packets lost in a single window of *window_length* buffers for each transfer and the number of transfers, out of fifteen, for which at least one service violation was reported to the PECC client.

The performance under the configurations in the first two rows of Table A.2 is quite similar to that for the configuration in the fourth row of Table A.1. That is, the window criteria in this case have little effect on the dynamic retransmission behavior. The information in the last two column of Table A.2 does indicates, however, that the number of buffers being skipped in any window is relatively few, and in fact the service constraint on errors is often met.

Variable Definitions:
dseq—one greater than the highest sequence number of data delivered to client.
rseq—one greater than the highest sequence number of data received in sequence.
pseq—sequence number of first byte of data in arriving packet.
Invoking Event:
Packet arrives at the PECC receiver with $pseq \neq rseq$.

```
        begin
0.          initialize curr_gap and win_credit; (19.1)
1.          if ( rseq − dseq ≥ fifo_min )
2.              send control packet for retransmission; (0.4)
3.          else (18.7)
4.              if ( curr_gap < min{max_gap, win_credit} )
5.                  skip curr_gap bytes; (10.3)
6.              else (8.4)
7.                  skip min{max_gap, win_credit} bytes;
8.                  if ( rseq − dseq ≥ fifo_min )
9.                      send control packet for retransmission; (8.3)
10.                 else
11.                     skip curr_gap − min{max_gap, win_credit} bytes;
12.                     record service violation; (0.1)
        end
```

Figure A.4: Trace of Average Behavior of PECC Algorithm.

To increase understanding of the behavior being observed, Figure A.5 presents trace statistics of the PECC algorithm under the configuration in the first row of Table A.2. The labels on the algorithm represent the number of times this line of the algorithm was execution in a single 1000-buffer transfer, as determined by averaging the trace statistics over all (fifteen) 1000-buffer transfers in the experiment.

The trace statistics indicate that the PECC client reads the XTP buffers fast enough that the buffers are often empty, e.g., line 2 in the algorithm is seldom executed. The current gap is found to be sufficiently small that it is skipped about 50% of the time, e.g.

line 5 is executed. When the current gap is too large to be skipped, it is very often the case that at least one buffer can be skipped (line 7). Since *fifo_min* is 1, this almost always allows for a resumption of reliable mode packet processing (line 9). If three buffers have already been skipped in the ten-buffer window, however, it will be the case that no extra buffers can be skipped with impunity at this point (lines 11 and 12).

We now explain the poor performance of the configuration in the third row of Table A.2. Since the *window_length* parameter is larger than in the first two configurations, the *win_credit* variable in the PECC algorithm has a value of 0 at line 7 more often than in the other two configurations of Table A.2. This reduces the likelihood that the test in line 8 is successful and thus increases the number of times that line 11 is executed. The result is a higher percentage of unrecovered data than in PECC configurations with a smaller window size.

A.6 Conclusions

In this appendix we have addressed the need for a new class of transport service for delay-sensitive applications. We have defined a Partially Error-Controlled Connection service that provides a very flexible interface for applications to parametrize the underlying delay-sensitive retransmission algorithm. We have demonstrated that the PECC service can be provided as a lightweight implementation enhancement to XTP. Measurements of experimental transfers show that the total transfer time on a PECC connection is similar to that for an XTP *no-error mode* connection and significantly less than for a fully reliable connection. Thus, through empirical measurements, we have demonstrated that a service intermediate between a reliable connection and the unreliable XTP *no-error mode* connection can be achieved. Under some configurations in our experiments, which use conditions of heavy loss, the PECC service is able to recover a high percentage (up to 70%) of the packets dropped by the network.

For applications that do not require 100% data completeness, selectively ignoring data loss, as done in a PECC channel, can improve the throughput of a transfer while retaining

the advantages of a connection-based service, e.g., data sequencing. The PECC service is of particular benefit to XTP multicast transfers. XTP defines an error-controlled multicast that uses go-back-n retransmission at the transmitter. Since an isolated packet loss at one receiver results in a retransmission of that packet for the entire multicast group, the limited number of retransmissions in a PECC service will have a significant impact on the throughput and delay performance of an XTP multicast. Since the XTP transmitter is unaware of the PECC mechanisms, the number of multicast receivers operating under the PECC rules can vary without an impact on processing at the multicast transmitter.

A possible drawback to the PECC approach is the difficulty that the application may have in choosing parameter values that result in the desired performance. The PECC parameters force the application to make judgements about the relative timing of network-level behavior. Bringing the application closer to the network is a general trend in real-time communication, but it does demand more of the application than in traditional communications. At least in the case of the PECC service, poor choices for parameter values yield at worst a *no-error mode* connection, i.e., no data will be recovered through retransmission. Since the PECC service is very lightweight, there is no processing performance penalty for the small amount of additional code executed by the protocol, though unsuccessful retransmissions may waste a small amount of network bandwidth.

Bibliography

[1] G. Barberis. Buffer Sizing of a Packet-Voice Receiver. *IEEE Transactions on Communications*, COM-29(2):152–156, February 1981.

[2] G. Barberis and D. Pazzaglia. Analysis and Design of a Packet-Voice Receiver. *IEEE Transactions on Communications*, COM-28(2):217–227, February 1980.

[3] V. Bhargava. Forward Error Correction Schemes for Digital Communications. *IEEE Communications Magazine*, pages 11–19, January 1983.

[4] E. Biersack. Performance Evaluation of Forward Error Correction in ATM Networks. *Computer Communications Review*, 22(4):248–258, August 1992.

[5] E. Biersack, C. Cotton, D. Feldmeier, A. McAuley, and W. Sincoskie. Gigabit Networking Research at Bellcore. *IEEE Network*, 6(2):42–48, March 1992.

[6] P. T. Brady. A Technique for Investigating On-Off Patterns of Speech. *Bell System Technical Journal*, 44:1–22, 1964.

[7] P. T. Brady. A Technique for Investigating On-Off Patterns of Speech. *Bell System Technical Journal*, 44(1):1–22, January 1965.

[8] P. T. Brady. Effects of Transmission Delay on Conversational Behavior on Echo-Free Telephone Circuits. *Bell System Technical Journal*, 50(1):115–134, January 1971.

[9] K. Brayer and S. Natarajan. An Investigation of ARQ and Hybrid FEC-ARQ on an Experimental High Altitude Meteor Burst Channel. *IEEE Transactions on Communication*, 37(11):1239–1242, November 1989.

[10] CCITT. Recommendation G.114.

[11] CCITT. Draft Recommendation I.321—B-ISDN Protocol Reference Model and its application, June 1990.

[12] D. Chase. Code Combining— A Maximum-Likelihood Decoding Approach for Combining an Arbitrary Number of Noisy Packets. *IEEE Transactions on Communication*, 33(5):385–393, May 1985.

[13] D. Cohen. Specification for the Network Voice Protocol (NVP). Technical Report RFC 741, Information Sciences Institute, Los Angeles, CA, January 1976.

[14] R. Cole. IP over ATM: A Framework Document, January 1994. Internet Draft.

[15] B. Dempsey, W. Strayer, and A. Weaver. Adaptive Error Control for Multimedia Data Transfer. *IWACA 1992*, pages 279–289, March 1992.

[16] R. Droms et.al. Report from the Joint SIGGRAPH/SIGCOMM Workshop on Graphics and Networking. *Computer Communication Review*, 21(2):17–25, 1991.

[17] S. Floyd and V. Jacobson. The Synchronization of Periodic Routing Messages. *Computer Communications Review*, 23(4):33–45, September 1993.

[18] J. Forgie and C. McElwain. Some Comments on NSC Note 78 'Effects of Lost Packets on Speech Intelligibility'. Technical Report Network Speech Compression Note 92, Lincoln Lab, Massachusetts Institute of Technology, March 1976.

[19] E. Fox. Advances in Interactive Digital Multimedia Systems. *IEEE Computer Magazine*, 24(10):9–21, October 1991.

[20] M.W. Garrett and M. Vetterli. Joint Source/Channel Coding of Statistically Multiplexed Real-Time Services on Packet Networks. *IEEE/ACM Transactions on Networking*, 1(1):71–81, February 1993.

[21] S. Golestani. Congestion-Free Communication in High-Speed Packet Networks. *IEEE Transactions on Communications*, 39(12):1802–1812, December 1991.

[22] F. Gong. *A Transport Solution for Pipelined Network Computing.* PhD thesis, Computer Science Department, Washington University at St. Louis, December 1992.

[23] F. Gong and G. Parulkar. An Application-Oriented Error Control Scheme for High-Speed Networks. Technical Report WUCS-92-37, Department of Computer Science, Washington University in St. Louis, November 1992.

[24] J. Gruber. Delay Related Issues in Integrated Voice and Data Networks. *IEEE Transactions on Communications*, COM-29(6):786–800, June 1981.

[25] J. Gruber. A Comparison of Measured and Calculated Speech Temporal Parameters Relevant to Speech Activity Detection. *IEEE Transactions on Communications*, COM-30(4):728–738, April 1982.

[26] V. Jacobson. VAT Protocol Specification, 1991. Lawrence Berkeley Laboratory, University of California, Berkeley.

[27] R. Jain. Performance Analysis of FDDI Token Ring Networks: Effect of Parameters and Guidelines for Setting TTRT. *IEEE Lightwave Telecommunications Systems*, pages 16–22, May 1991.

[28] N. Jayant. High Quality Networking of Audio-Visual Information. *IEEE Communications*, 31(9):84–95, September 1993.

[29] N. Jayant and S. Christensen. Effects of Packet Losses on Waveform-Coded Speech and Improvements Due to an Odd-Even Interpolation Procedure. *IEEE Transactions on Communication*, COM-29(2):101–109, February 1981.

[30] N. Jayant and P. Noll. *Digital Coding of Waveforms.* Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[31] M.J. Johnson. Proof that Timing Requirements of the FDDI Token Ring Protocol Are Satisfied. *IEEE Transactions on Communications*, COM-35:620–625, June 1987.

[32] S. Kallel. Analysis of a Type II Hybrid ARQ Scheme with Code Combining. *IEEE Transactions on Communication*, 38(9):1133–1137, August 1990.

[33] G. Karlsson and M. Vetterli. Packet Video and Its Integration into the Network Architecture. *IEEE Journal on Selected Areas in Communications*, 7(3):739–751, June 1989.

[34] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in Reliable Transport Protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, November 1991.

[35] E. Klemmer. Subjective Evaluation of Transmission Delay in Telephone Conversations. *Bell System Technical Journal*, 46:1141–1147, July 1967.

[36] J. Kurose. Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks. *ACM Computer Communication Review*, 23(1):6–15, January 1993.

[37] C. Leung and A. Lam. Forward Error Correction for an ARQ Scheme. *IEEE Transactions on Communication*, COM-29(10):1514–1519, October 1981.

[38] M. Liou. Overview of the p-x-64 kbit/s Video Coding Standard. *Communications of the ACM*, 34(4):59–63, April 1991.

[39] A.J. McAuley. Reliable Broadband Communication Using a Burst Erasure Correcting Code. *Computer Communications Review*, 20(4):297–306, September 1990.

[40] J. McNabb. WIRETAP User's Manual, 1993. Internal Report, Department of Computer Science, University of Virginia.

[41] D. Mills, December 1983. Internet RFC 889.

[42] W.A. Montgomery. Techniques for Packet Voice Synchronization. *IEEE Journal on Selected Areas in Communications*, SAC-1(6):1022–1028, December 1983.

[43] M. Moran. Design of a Continuous Media Data Transport Service and Protocol. Technical Report TR-92-019, International Computer Science Institute, University of California Berkeley, April 1992.

[44] A. Mukherjee. On the Dynamics and Significance of Low Frequency Components of Internet Load. Technical Report CIS-92-83, University of Pennsylvania, December 1992.

[45] W.E. Naylor and L. Kleinrock. Stream Traffic Communication in Packet-Switched Networks: Destination Buffering Considerations. *IEEE Transactions on Communications*, COM-30(12):2527–2534, December 1982.

[46] A. Netravali, W. Roome, and K. Sabnani. Design and Implementation of a High-Speed Transport Protocol. *IEEE Transactions on Communications*, 38(11):2010–2024, November 1990.

[47] H. Ohta and T. Kitami. A Cell Loss Recovery Method using FEC in ATM Networks. *IEEE Journal on Selected Areas in Communications*, 9(9):1471–1483, December 1991.

[48] A. Sanghi, A. Agrawala, and B. Jain. Experimental Assessment of End-to-End Behavior on the Internet. *IEEE INFOCOM '93*, pages 867–874, March 1993.

[49] T. Sato, M. Kawabe, T. Kato, and A. Fukasawa. Throughput Analysis Method for Hybrid ARQ Schemes Over Burst Error Channels. *IEEE Transactions on Vehicular Technology*, 42(1):110–118, February 1993.

[50] H. Schulzrinne. Voice Communication Across the Internet: A Network Voice Terminal. Technical report, University of Massachusetts, July 1992.

[51] SES. SES Workbench Release 2.1, February 1992. Scientific and Engineering Software.

[52] N. Shacham and P. McKenny. Packet Recovery in High-Speed Networks using Coding. *INFOCOM 1990*, pages 124–131, June 1990.

[53] A. Shah, D. Staddon, I. Rubin, and A. Ratkovic. Multimedia over FDDI. *17th IEEE Conference on Local Computer Networks*, pages 110–124, September 1992.

[54] R. Simoncic, A. Weaver, and A. Colvin. Experience with the Xpress Transfer Protocol. *15th IEEE Conference on Local Computer Networks*, pages 123–132, September 1990.

[55] K. Sriram and W. Whitt. Traffic Smoothing Effects of Bit Dropping in a Packet Multiplexer. *IEEE Transactions on Communications*, 37(7):703–712, July 1989.

[56] W. Stallings. *Data and Computer Communications*. Macmillan Publishing Company, 1991.

[57] W. Strayer, B. Dempsey, and A. Weaver. *XTP: The Xpress Transfer Protocol*. Addison-Wesley Publishing, July 1992.

[58] J. Suzuki and M. Taka. Missing Packet Recovery Techniques for Low-Bit-Rate Coded Speech. *IEEE Journal on Selected Areas in Communications*, 7(5):707–717, June 1989.

[59] T. Turletti. H.261 Software Codec for Videoconferencing over the Internet. Technical Report 1834, Institut National de Recherche en Informatique et en Automatique (INRIA), January 1993.

[60] T. Turletti, February 1994. personal communication.

[61] D. Verma, H. Zhang, and D. Ferrari. Guaranteeing Delay Jitter Bounds in Packet-Switching Networks. *Tricomm '91*, April 1991.

[62] S. Webber, C. Harris, and J. Flanagan. Use of Variable-Quality Coding and Time-Interval Modification in Packet Transmission of Speech. *Bell System Technical Journal*, 56:1569–1573, October 1977.

[63] N. Yin. Congestion Control for Packet Voice by Selective Packet Discarding. *IEEE Transactions on Communications*, 38(5):674–683, May 1990.

[64] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. *Computer Communications Review*, 21(4):113–211, September 1991.