Stream Synchronization for Voice over IP Conference Bridges

Colm Elliott

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

November 2004

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

Potentially large network delay variability experienced by voice packets when travelling over IP networks complicates the design of a robust voice conference bridge. *Intra*stream synchronization is required at the conference bridge to smooth out network delay jitter on a given stream and provide a continuous stream of voice packets to the conference bridge core. *Inter*stream synchronization is needed to provide time synchronization between packets in different streams, allowing for a mapping of selected voice streams to the conference bridge output and the creation of a periodic and synchronized output from the conference bridge.

This work presents a design and evaluation of a Synchronized conference bridge that maps N input voice streams to M output voice streams representing selected speakers. A conference simulator, designed for this thesis, is used to characterize the performance of this bridge in terms of delay and packet loss, speaker selection accuracy and conference audio quality.

Sommaire

Les grandes variations de délai que subissent les paquets audios durant une transmission à travers un réseau IP compliquent la tâche d'assurer un système robuste de téléconférence de parole centralisé. En effet, une méthode de synchronisation "intrastream" est requise afin d'obtenir un flot continue de paquets audio au cœur du pont de téléconférence. Une méthode "interstream" parvient une synchronisation entre les paquets appertenant à différents flots vers la sortie du pont de conférence et dans un deuxième temps, la création d'une sortie périodique et synchronisé.

Ce mémoire présente la conceptualisation ainsi que l'évaluation d'un pont de téléconférence synchronisé que effectue l'application de N flots audio d'entrée vers M flots audio de sortie représentants les conférienciers désignés. Un simulateur de téléconférence est aussi élaboré afin d'aider à caractériser la performance de notre pont en terms de délais et perte de paquets, de précision en ce qui concerne la sélection de conférenciers et enfin, de la qualité audio de la téléconférence.

Acknowledgments

I would like to express my thanks to my supervisor, Professor Peter Kabal, for his guidance, patience, and generous financial support. I am especially thankful to Paxton Smith, for advice, support, and feedback during the course of my research, as well as for providing test conference audio. Thanks go out to Aziz Shallwani for providing network delay traces and to Karim Ali for help with the abstract translation.

I am grateful to my friends for their support and for times shared over the last two years. A special thanks to Jenny for her patience and faith in me, and to my parents for their support.

Contents

List of Figures

List of Tables

List of Terms

ACELP	Algebraic-Code-Excited Linear-Prediction
AGC	Automatic Gain Control
AR	Autoregressive
CNG	Comfort Noise Generation
DTX	Discontinuous Transmission
FEC	Forward Error Correction
IETF	Internet Engineering Task Force
IP	Internet Protocol
ITU-T	International Telecommunications Union
MOS	Mean Opinion Score
NLMS	Normalized Least Mean Square
NTP	Network Time Protocol
PCM	Pulse Code Modulation
PLC	Packet Loss Concealment
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RTP	Real Time Transport Protocol
RTCP	RTP Control Protocol
SID	Silence Insertion Descriptor
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VAD	Voice Activity Detector
VoIP	Voice over IP
WSOLA	Waveform Synchronized Overlap-Add

Chapter 1

Introduction

Conferencing capability allows for group communication and collaboration among geographically dispersed participants and forms an integral part of any voice communication network. Historically, conferencing has been achieved in the Public Switched Telephone Network (PSTN) by means of a centralized conference bridge.

Currently trends point towards the migration of voice communication services from the circuit-switched PSTN to packet-based Internet Protocol (IP) networks. This shift is motivated by a desire to provide data and voice services on a single, packet-based network infrastructure. This integration of voice and data allows for cost reduction and the ability to provide rich multimedia services that combine voice, video, text and data.

While the PSTN was designed for real-time voice communications, IP networks were designed for the transport of packetized data. Several challenges must be overcome in order to provide a Voice over IP (VoIP) solution that is comparable to the PSTN in terms of reliability and voice quality. Voice packets that are independently routed across an IP network result in variable packet delays and the potential for misordered delivery of packets as well as packet loss. Playout buffering of packets is required at VoIP endpoints in order to reconstruct a continuous voice stream.

Migrating voice *conferencing* solutions from the PSTN to IP networks presents even more challenges than in the case of a one-to-one conversation. Variable network delays *between* voice streams must be handled, in addition to delay variability found within a given stream.

1.1 PSTN Conferencing

Public Switched Telephone Network (PSTN) conferencing is usually accomplished with a conference bridge, which sums the input signals of each conferee before returning the summed signals to endpoints [1].

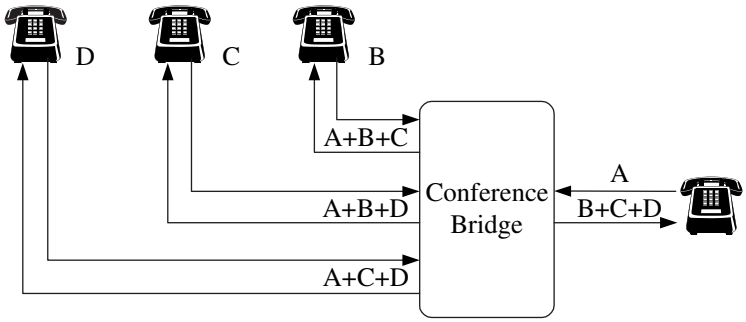


Fig. 1.1 PSTN Conference Bridge. The bridge forms a unique sum for all conferees [2].

Voice circuits are "nailed-up" between conference endpoints and the bridge, providing a dedicated low-delay circuit-switched connection that experiences network jitter on the order of microseconds [2]. The use of low-complexity waveform coders, such as G.711, places a relatively small processing load on bridges when decoding incoming voice streams for summation, and injects little appreciable distortion when voice signals are subject to tandem encodings during the decode-mix-encode operation.

PSTN conference bridges can buffer incoming streams by an amount equal to or greater than the maximum jitter experienced over the network link, providing a continuous stream a speech samples to the bridge core, and effectively eliminating any synchronization issues within or between voice streams. Since the maximum jitter is so small, this buffering delay will not result in any appreciable degradation of service to end users.

1.2 IP Conferencing

Fig. 1.2 shows a generic VoIP conference bridge, as presented in [3]. In order to mix incoming streams, packets are buffered so as to remove network delay jitter and attempt to provide a continuous stream of packets to the bridge's mixing module. Because network delay jitter can be high on wide-area IP networks and packets may be lost over the network,

buffering incoming packets by an amount equal to the maximum delay jitter is not a feasible solution. This would result in a high end-to-end delay that would significantly degrade the conversational quality. Buffering of input streams in IP networks becomes a trade-off between delay and late arriving packets.

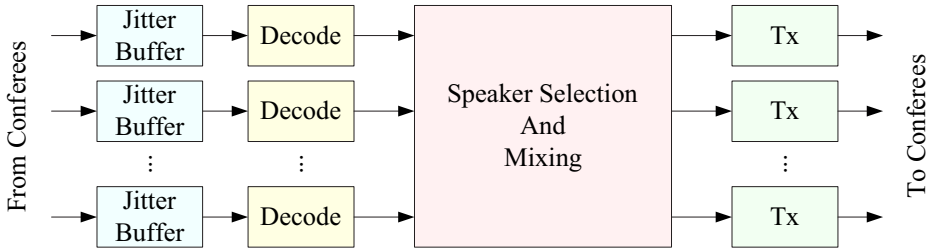


Fig. 1.2 Generic VoIP Conference Bridge [3].

Speech arrives at an IP bridge in units of packets, rather than on a sample-by-sample basis as in the PSTN. Speech output from the bridge is also in units of packets. Due to the inevitability of lost or late-arriving packets at an IP bridge, a continuous stream of samples can not be guaranteed as input to the bridge core and the mixing operation is simplified if done on a packet-by-packet basis. An *inter*stream synchronization mechanism is needed to map and time synchronize packets from one voice stream to the appropriate packets in another stream, for the purposes of mixing or bundling selected packets.

1.2.1 Tandem-Free Conferencing

While the PSTN has universal adoption of a common codec (G.711), conference endpoints may use a wide array of speech codecs. Tandem encodings of high-compression codecs such as G.723.1 and G.729 result in degradation of speech quality, as does the encoding of a multi-talker signal, as these codecs are optimized for a single talker. Tandem-free conference bridges avoid tandem encodings by forwarding selected speaker's packets and offloading the decode-mix operation to endpoints [4].

A tandem-free "Select-and-Forward" conference bridge is presented in [5]. Speaker selection is performed on unbuffered input voice streams, and packets selected as speakers are forwarded to conference endpoints, where packets are decoded and mixed. While this approach avoids tandem encodings and does not inject any buffering delay at the bridge, conference endpoints are not *abstracted* from conference details — they must be aware of all other conference endpoints and support a virtual connection with each of them.

The Synchronized bridge presented in this paper also operates tandem-free. Packets are not mixed at the bridge, but bridge output streams are time synchronized and periodic. This allows for easy extensibility to bundling or mixing of selected packets.

1.3 Thesis Contribution

This thesis provides a design and implementation of a Synchronized conference bridge suitable for IP networks. Edholm *et al.* present a design for an IP conference bridge in [3], but do not provide any details of mechanisms that would allow for the synchronization of incoming streams for the purposes of mixing. Work presented here details novel bridge synchronization algorithms that allow the bridge to map N voice streams originating from conference endpoints, to M output streams representing selected speakers, which are then sent back to conference endpoints. This mapping allows for the bridge to abstract network delays incurred by packets on the source-to-bridge path, and to abstract the packet source. The design presented in this thesis operates tandem-free, but the bridge synchronization mechanism allows for easy extensibility to mixing of selected streams.

A prototype of the Synchronized bridge using these novel synchronization algorithms was developed and evaluated using a conference simulator, also developed for this thesis. A prototype of the Select-and-Forward bridge presented in [5] was also implemented and evaluated. The comparative evaluation of the Select-and-Forward and Synchronized bridges allowed for the characterization of the delay penalty attributable to performing stream synchronization at the bridge. Complimentary performance metrics, such as voice synchronization and speaker selection accuracy, allowed for further evaluation of the two bridges. Output audio generated by the conference simulator provided additional characterization of performance.

1.4 Thesis Organization

Chapter 2 lays out the fundamentals of VoIP as well as presenting the basics of centralized conferencing. A brief survey of network protocols involved in VoIP is followed by a discussion of VoIP endpoint components, with special focus on speech codecs. Intrastream synchronization, or playout scheduling, is then examined in detail, as are algorithms for interstream synchronization. The discussion on centralized conferencing touches speaker

selection, mixing, and the implications of using high-compression speech codecs, as well as presenting a detailed description of the Select-and-Forward bridge.

The Synchronized bridge design is presented in Chapter 3. Algorithms for intrastream and interstream synchronization are described, as are the particulars of packet header translation and management of late packets.

Evaluations of conferences controlled by both the Synchronized and Select-and-Forward bridges are presented and compared in Chapter 4. Multiple network delay configurations are used for conference simulations, and bridges are evaluated based on delay, packet loss, voice synchronization, speaker selection accuracy, and voice quality. The effects of packet length and clock skew between endpoint sampling clocks are also examined. A discussion on the relative complexity and scalability of conferencing environments controlled by each of the two bridge types ends the chapter.

A summary of results as well as suggestions for potential improvements and future research are presented in Chapter 5.

Chapter 2

Voice over IP Conferencing

Transporting voice over the Internet presents various challenges, including dealing with network delay, network delay jitter, and packet loss. Providing a *voice conferencing* solution for IP networks entails even more challenges, as voice streams need to be combined to form a true multi-way conversation. A voice *conference* is defined here as an N-way conversation, with $N > 2$.

The variability in packet delays within a given voice stream as well as those found between different voice streams provide challenges to the centralized conferencing problem that are not found in the equivalent solution for the PSTN. Voice codecs used in VoIP applications require additional consideration because of voice quality degradation under tandem encodings and extra complexity introduced by state dependant codecs. Unlike in the PSTN where one can assume the use of the same low complexity voice encoding from all participants, the potential for different conference participants using different voice codecs with different packetization intervals must also be considered.

This chapter gives an overview of the challenges inherent to VoIP and explains some of the current approaches used to deliver voice over IP networks, as well as giving background on the mechanisms of centralized conferencing. Section 2.1 gives a brief overview of IP networks, while Section 2.2 introduces the Real Time Transport Protocol (RTP), the protocol of choice for delivering real time media over packet networks. Section 2.3 discusses some challenges in delivering voice over the internet as well as describing the basic components of a VoIP receiver. Section 2.4 focuses on intrastream synchronization, in more detail, and discusses various approaches used to deal with network delay jitter. The

topic of interstream synchronization is discussed in Section 2.5, dealing with timing issues between temporally related media streams. Section 2.6 introduces conferencing bridges and their components, as well as challenges inherent to centralized conferencing over IP networks. Section 2.7 explores tandem-free operation and provides a detailed description of a Select-and-Forward bridge [5]. Finally, Section 2.8 discusses other possible conferencing topologies and approaches to conferencing over the internet.

2.1 IP networks

Internet Protocol (IP) is a connectionless protocol residing at the network layer (layer 3) of the Open Systems Interconnection (OSI) reference model, responsible for routing information over the network. The IP layer offers *best effort* delivery of packets and as such provides no explicit Quality of Service (QoS) guarantees, sequencing, flow control or acknowledgements [6]. Some or all of these reliability functions can be provided at the transport layer, which provides end-to-end communication control. IP is a packet-based protocol where the payload to be transported is split up into separate datagrams which are independently sent across the network. The IP layer provides no delay bounds on the transmission time of a packet, nor does it guarantee that packets will be received in order.

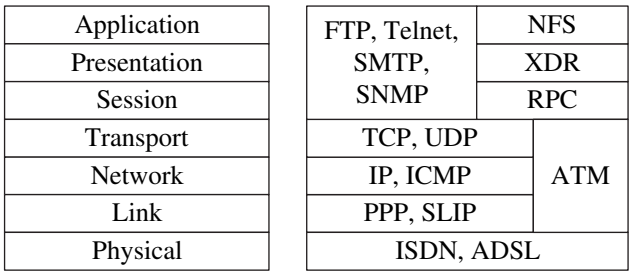


Fig. 2.1 OSI reference model. Examples of protocols residing at each layer are given [6].

2.1.1 Addressing Mechanisms

IP packets can be addressed and delivered by one of three means: unicast, broadcast, or multicast. Unicast is used for simple point-to-point communication between two hosts (nodes or computers on a network). Broadcast allows for a packet to be sent to all hosts

on a given subnetwork using only one logical address. Multicasting allows for the delivery of packets from one host to multiple other hosts in a bandwidth efficient manner: only one copy of each packet is sent from the multicast source and the packet is replicated as needed in network nodes during delivery. Multicast requires support from network routers, and as such does not have full availability or adoption [7]. The one-to-many functionality provided by multicast is well suited to large conferences or webcasts. The less bandwidth efficient alternative to multicast, known as multi-unicast, is for the source host to generate a copy of its output stream for each receiving host.

2.1.2 Transport Protocols

Transport layer protocols can provide reliability guarantees for end-to-end transport across an IP network. The two most commonly used transport layer protocols on top of IP are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Transmission Control Protocol (TCP) is used for most Internet traffic that is not delay sensitive. It is a connection-oriented protocol, in that it provides flow control between two hosts as well as sequencing mechanisms to ensure the ordered delivery of data. It provides reliable delivery of IP packets through the use of packet acknowledgements and retransmission of lost packets. User Datagram Protocol (UDP) is a much lighter weight transport layer protocol and provides neither packet acknowledgements nor mechanisms for retransmission of lost packets. It does provide integrity checking with the use of checksums, but serves mostly to multiplex/demultiplex IP traffic [7].

Real time applications such as VoIP are not suited to run over TCP as the delays introduced by retransmissions of packets are too long to allow for interactive and/or time sensitive applications. UDP is thus the transport protocol of choice for real time applications such as VoIP, where a certain amount of packet loss is preferable to additional network delays incurred by acknowledgement and retransmission schemes such as those provided by TCP.

2.2 Real Time Transport Protocol

Real time media applications generally use the Real Time Transport Protocol (RTP), which runs on top of UDP and provides the necessary timing and packet sequencing information to allow for the reconstruction of a ordered and continuous real-time data stream at the

receiver. Any packet loss is dealt with (or absorbed) by a higher level application. While Real Time Transport Protocol (RTP) does not itself provide any QoS guarantees, it does provide mechanisms for a higher level application to do so.

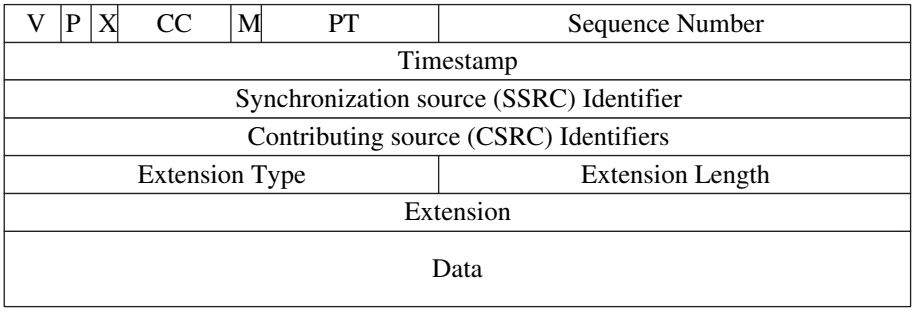


Fig. 2.2 RTP packet format. V = version, P = padding flag, X = extension flag, CC = number of contributors in CSRC field, M = marker bit, PT = payload type. CSRC field and Extension are optional [8]

The format of an RTP packet is shown if Fig. 2.2. The RTP *sequence number* is a 16 bit integer which increments by one for each RTP packet sent. The sequence number allows the receiver to detect packet loss and reorder packets that may have arrived out of sequence. The initial value of the sequence number (i.e. the sequence number assigned to the first packet in the stream) is chosen at random, so as to add robustness in the face of known plaintext attacks in the advent that encryption is being used [8].

The RTP *timestamp* is a 32 bit integer specifying the sampling instant of the first byte of data in the RTP data packet [8]. For audio streams, this timestamp is tied to the source sampling clock. The RTP timestamp is incremented by the number of sampling clock instants in a packet for each successive packet. For example, for 20ms voice packets containing 160 samples each, the timestamp will increment by 160 for each packet sent. As in the case of the sequence number, the initial value of the timestamp is chosen randomly for security reasons. The timestamps allow the receiver to estimate the network delay variation, or *jitter*. The relative network delay of the ith packet can be calculated by subtracting the timestamp field of the incoming packet (t_i), from the value of a counter tied to the local sampling clock at the time of packet reception (T_i).

$$n_i = T_i - t_i \tag{2.1}$$

The relative network delay, n_i, given in units of samples, is equal to the actual network delay plus an offset between the source and receiver sampling clocks. This offset will remain constant for the duration of the RTP session (neglecting clock skew — see Section 2.3.7). Subtracting the relative network delay for any two packets will give the difference in the actual network delay of the two packets, allowing for the calculation of a relative network mean, and actual network delay variation. This will be discussed in further detail when discussing playout scheduling in Section 2.4.

The *data* field is the actual data payload (coded voice samples in the case of VoIP). Other mandatory fields included in an RTP header are the *payload type*, which identifies the format of the RTP data, and the Synchronization Source (SSRC) field, which identifies the media stream. The optional CSRC field identifies sources that contributed to the payload in cases where data from multiple sources is mixed together (as in centralized conferencing where the bridge performs mixing) [8].

2.2.1 RTP Mixers and Translators

There are two notions of intermediate systems supported by RTP: mixers and translators. A *translator* is an intermediary through which an RTP packet passes without having its SSRC identifier changed, while the packet sequence number and/or timestamp may or may not be changed. An example of a translator is a Select and Forward bridge (Section 2.7.1). In general, receivers will not be able to detect the presence of translators, as the packet will still seem to come from its original source [8].

A mixer is an entity that receives RTP streams from one or more sources, combines them in some way, and forwards the new mixed stream to one or more receivers. The mixer will need to time synchronize the incoming streams and generate its own timestamps and sequence numbers. A mixer must change the SSRC field such that it appears as the source of the new combined packet, while identifiers for the original incoming streams that were mixed should be included in the CSRC field. An example of a mixer is a Synchronized bridge (Section 3). Packets arriving from a mixer will show the mixer as the *source* of the packet. As a source, the mixer must ensure that its output RTP stream exhibits valid timing behaviour: sequence numbers in outgoing packets must be incremented by one and timestamps must be incremented by the number of sampling instants since the last outgoing packets.

2.2.2 Logical Streams and Logical Sources

The terms *logical stream* and *logical source* will be used extensively when discussing the technical details of mixing and/or modifying media streams. A *logical stream* is a flow of packets in which the payload makes up a continuous media stream. A logical stream is characterized by the source identifier (SSRC) on RTP packet headers; packets with the same SSRC belong to the same logical stream. A *logical source* is is characterized by the network delay characteristics of packets travelling between it and a destination. Two logical streams incoming to a given destination that have identical (or at least very similar) network delay characteristics based on their RTP timestamps are considered to come from the same logical source. As an example, the transmission of a video stream and an audio stream from source A to destination B would constitute two *logical streams* from one *logical source*. Each logical stream needs to be buffered (i.e. dejittered) separately, but logical streams from the same logical sources can share network delay information on which to base playout scheduling.

2.2.3 RTP Control Protocol

RTP Control Protocol (RTCP) is a control protocol that provides feedback on the quality of the corresponding RTP session. RTCP runs in parallel with RTP, also on top of UDP, but using a different port. In addition to quality feedback reports, RTCP can provide absolute wallclock timestamps that allow the receiver to map actual time to the sender's RTP timestamps. RTCP can also optionally provide minimal session control information. RTCP packets are sent periodically, with the sending interval dependent on the size of the session (number of participants) and bandwidth limitations. A minimum interval between RTCP packets of 5 seconds is suggested [8].

Feedback provided in RTCP reports includes the number of packets received, number of packets lost, and an estimate of the interarrival jitter. If the source is running Network Time Protocol (NTP), an NTP timestamp, giving absolute date and time (in units of seconds since the 1st of January, 1900) is included in the RTCP packet, along with the RTP timestamp corresponding to that time. If the receiver is also running NTP, the sender RTP timestamp can be mapped to the receiver sampling clock, effectively determining the offset between the two sampling clocks. This allows the receiver to estimate the actual network delay of received packets (with an error dictated by the time granularity of NTP) instead of

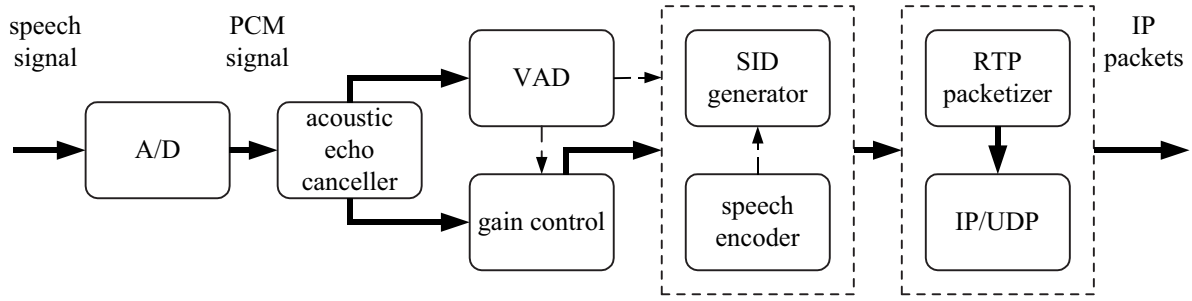
just a relative network delay, as calculated in Eq. (2.1). While feedback provided in RTCP reports may be used as input parameters for playout scheduling algorithms (Section 2.4) or adaptive speech coding (Section 2.3.1), their main function is to diagnose faults in the distribution and network [8].

2.3 VoIP Endpoints

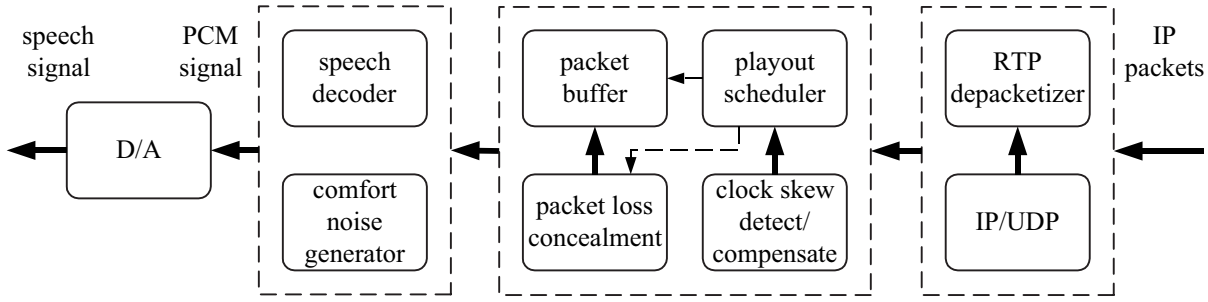
In order to deliver quality real-time voice signals over IP networks, multiple challenges must be met. First, the source must digitize, encode and packetize the speech signal to be transported. This encoding becomes more complex if low bandwidth utilization is required, and may involve the use of VAD and a silence suppression/comfort noise generation scheme. Network delay variations must be absorbed and smoothed out at receivers so as to provide continuous playout of voice packets. Receivers and/or sources must provide mechanims to mitigate the perceptual degradation of speech due to lost packets. The relatively large round trip delays and the large variability in round trip delay incurred by voice packets over the Internet, (as compared to the PSTN) require additional echo cancellation mechanisms. A generic VoIP endpoint is shown in Fig. 2.3.

2.3.1 Speech Coding

The aim of a speech coder, or codec, is to achieve an efficient digital representation of a speech signal that is suitable for transmission over a network. Analog speech signals are digitized using a form of Pulse Code Modulation (PCM). Analog voice is typically sampled at 8 kHz and bandpass filtered from 300–3400 Hz, as most speech energy resides in that frequency range [9]. Linear PCM samples are then sent to the coder, which constructs a compressed representation of theses samples. For packet networks such as IP networks, frames of speech (typically 10–40ms or 80–320 samples) are then packetized for transmission and sent across the network. The receiver decodes the incoming packets, and the resulting PCM samples are converted back to an approximation of the original analog speech signal. In general, the process of encoding and decoding of speech will result in a loss of information. Low bit rate codecs can achieve transmission rates as low as 2.4 kbps (or less than 0.5 bits/sample) while standard A-law and μ-law codecs used in the PSTN have rates of 64 kbps (or 8 bits/sample) [7]. In general, the quality of the decoded speech will decrease as the bit rate is reduced, but codecs that take advantage of redundancies found in speech



(a) A VoIP transmitter.



(b) A VoIP receiver.

Fig. 2.3 A generic VoIP transmission system [5].

signals can achieve toll quality speech with a large savings in bit rate, usually at the price of increased coder complexity.

Speech coders come in three main categories: waveform coders, source coders (vocoders), and hybrid coders. Waveform coders attempt to represent the analog signal directly. Source coders use a set of parameters that drive a speech production model, such as a digital filter that models the speaker's vocal tract [7]. Hybrid coders, as the name might suggest, use a combination of the aforementioned approaches.

Speech coders are evaluated on several criteria including bit rate, speech quality, algorithmic complexity, lookahead delay, and frame size. Other qualities that are of special interest to VoIP and/or conferencing are robustness to tandem encodings, coding quality of multi-speaker signals, and robustness to packet loss.

Speech quality is generally measured using the Mean Opinion Score (MOS) scale, a subjective benchmark with ratings going from 1 (bad) to 5 (excellent). International Telecom-

munications Union (ITU-T) Recommendation G.113 [10] can also be used to to quantify the perceptual impairment to speech quality introduced by a given speech coder, given by an equipment impairment rating I_e. G.113 is part of the E-model (ITU-T G.107 [11]), which attempts to model an entire transmission path using parameters such as packet loss, network delay, echo, speech codecs and more to give an overall perceptual rating, R, ranging from 0 (bad) to 100 (excellent). The R rating is calculated by starting with a nominal R value of 94.3, and then subtracting *impairment* values attributable to factors such as equipment, delay, packet loss and echo (in some cases the R value will be increased for perceived bonuses such as mobility) [11].

The frame size of a codec refers to the smallest unit of speech that can be transmitted; some codecs code speech on a sample by sample basis while others consider a block of speech samples at a time. A larger frame size will in general add delay to the transmission path, as the time required to process a frame at the decoder is assumed to be the same as the frame length [12]. A larger frame also limits flexibility in choosing the RTP packetization interval (it must be a multiple of the frame size), which can limit performance of some centralized conferencing architectures.

Lookahead delay refers to the amount of the next frame that must be looked at in order to code the present frame. This is of importance, as it will result in additional delay for the voice packet, as well as potentially complicate switching between different voice streams, as done in some conferencing setups [13].

In general, lower bit rate codecs will have higher complexity and/or lookahead delays and/or lower speech quality. Most codecs have a state: the decoding of a given voice packet requires information from the previous packet(s). Lost packets will in these cases result in a loss of synchronization between encoder and decoder, and may cause errors in packets arriving after a lost packet. State synchronization requirements can also complicate RTP translators that multiplex and/or mix several voice streams (see Section 2.6.5).

Waveform Coders

Waveform coders generally offer high quality speech, low complexity and no lookahead delay, but at the cost of high bit rates. ITU-T G.711 [14], the well-known codec often associated with the PSTN, uses logarithmic compression on 13 (A-law) or 14 (μ-law) bit linear PCM samples to represent each sample with only 8 bits. G.711 is stateless and codes

each sample individually so is capable of arbitrary packetization intervals. It is also robust in the face of tandem encodings and provides for easy inter-operation with the PSTN as there is no need for speech format conversion. While G.711 may be an attractive option for high bandwidth environments, its 64 kbps bit rate is considered high for many VoIP applications.

Adaptive Differential Pulse Code Modulation (ADPCM) coders are another class of waveform coders that can achieve lower bit rates than G.711. This is done by coding the difference between the sample and a predicted value of that sample (based on the previous samples), and dynamically adapting the range of the quantization steps. ITU-T G.726 [15] describes ADPCM codecs that operate at bit rates of 16, 24, 32 and 40 kbps. While ADPCM codecs still work on a sample by sample basis (i.e. there is no frame delay), the decoding of a given samples depends on the decoded values of previous samples, thus requiring state. Lost packets will thus require state resynchronization.

Source Coders

Source coders, otherwise known as parametric coders or vocoders, generate a set of parameters to drive a model for speech production. Source coders can achieve bit rates as low as 2.4 kbps, but suffer from a poorer quality, synthetic sounding voice signal that is usually deemed unacceptable for toll-quality applications. Performance degrades rapidly in the face of tandem encodings or coding of multiple speaker signals. Source coders are generally only used when subjected to very tight bandwidth requirements and are thus not commonly used in most VoIP applications [7].

Hybrid Coders

Hybrid coders also transmit parameters for a synthesis filter, but also provide an excitation (or excitations) to this filter. The excitation is chosen from a codebook so as to minimize the error between the actual speech and the synthesized excitation. Hybrid coders consider a frame of speech at a time (typically 10–30ms), and thus can only use packetization intervals that are a multiple of the frame size. While having high complexity, hybrid coders provide toll quality speech at relatively low bit rates (5.3–16 kbps), making them attractive for many VoIP applications. Many hybrid coders have a lookahead delay as the coder needs to examine a portion of the next frame of speech in order to code the current frame.

Two prevalent hybrid coders for VoIP are ITU-T G.729 [16] and ITU-T G.723.1 [17]. G.729 operates on 10ms frames at 8 kbps with a lookahead delay of 5ms. A lower complexity implementation, G.729A, provides slightly lower quality speech with a 50% reduction in complexity. G.723.1 operates on 30ms frames at 6.3 kbps (Multipulse Maximum Likelihood Quantization (MP-MLQ) or 5.3 kbps (Algebraic-Code-Excited Linear-Prediction (ACELP)) with a lookahead delay of 7.5ms and delivers slightly lower quality speech than G.729. The relatively large frame size of G.723.1 makes it less flexible in the choice of RTP packet size (one is constrained to multiples of 30ms). G.723.1 is often used in video with voice applications as the video coding delay is usually large anyway, making the large frame size less of a constraint [7]. While G.729 and G.723.1 provide large savings in bit rate with a relatively small degradation in speech quality, they are high in complexity and highly state dependant. They are also subject to speech quality degradations in the face of tandem encodings and periods of multi-talk [9].

Table 2.1 Commonly used speech codecs. MOS values here were taken from [18] and [9] - codecs listed in both had corroborating values. The I_e rating is assumed to be additive when two codecs are used in tandem, although this claim has not been subject to extensive testing under many codec combinations [11]. Delay values are the sum of the algorithmic (frame) delay and lookahead delay.

Standard	Method	Bitrate (kbps)	Delay (ms)	Complexity (MIPS)	Quality (MOS)	Impairment (I_e)
G.711	LOG PCM	64	0.125	0.01	4.1	0
G.726	ADPCM	40	0.125			2
G.726		32	0.125	2	3.85	7
G.726		24	0.125			25
G.726		16	0.125			50
G.728	LD-CELP	16	0.675	30	3.61	7
G.729	CS-ACELP	8	15	20	3.92	10
G.729 x2					3.27	20
G.729 x3					2.68	30
G.729A	CS-ACELP	8	15	10.5	3.7	
G.723.1	MP-MLQ	6.3	37.5	14.6	3.9	15
G.723.1	ACELP	5.3	37.5	7.5	3.65	19

2.3.2 Voice Activity Detection

The goal of a Voice Activity Detector (VAD) is to assign a binary value to a frame of speech based on its classification as speech (1) or silence (0). For VoIP applications, a Voice Activity Detector (VAD) decision usually corresponds to the classification of the current packet (usually 10–40ms). VADs classify a segment as speech or silence based on a comparison between the average signal energy and a noise threshold [19, 20]. Most VADs employ a *hangover*, meaning that frames identified as silence immediately following frames classified as speech are also classified as speech. This prevents *clipping* and smooths speech to silence transitions by filling in spaces between words and sentences. The frames of speech are classified into a series of 1's and 0's representing talkspurts and silence periods [19].

The detection of silence periods allows for the use of *silence suppression*, or Discontinuous Transmission (DTX). Packets classified as silence need not be transmitted, providing bandwidth savings in the range of 50–60% for two-way conversations and even more in conferencing scenarios [21]. In most silence suppression schemes, packets are still sent intermittently during silence periods. These Silence Insertion Descriptor (SID) packets allow for the parametrizing of background noise and allow the receiver to update its network delay estimates. The receiver uses the parametrized background noise in the SID frames to update its Comfort Noise Generation (CNG). Comfort noise is played out during silence so the line doesn't sound like it has gone dead [22].

2.3.3 Playout Buffers

Voice packets will experience varying delays when travelling from source to receiver over an IP network. Although generated at regular intervals, packets will arrive at the receiver at irregular intervals and may even be lost or arrive out of order. In order to reconstruct the packet stream for continuous (or near-continuous) playout at the receiver in the face of variable network delay variability (i.e. delay jitter), incoming packets are buffered such that a majority of packets will arrive in time for continuous playout. The amount of time a packet is buffered, known as the buffering delay, is a trade-off between the rate at which packets will arrive too late for playout (and thus effectively be lost) and additional delay incurred due to playout buffering. Audio stream loss rates of up to 5% are generally considered tolerable if packet loss concealment algorithms are employed [23]. Interactivity of conversations tends to degrade as the end-to-end delay surpasses 150ms, while remaining tolerable up to delays

of 400ms [12]. Playout scheduling (also known as *intra*stream synchronization) algorithms will be discussed in more detail in Section 2.4.

2.3.4 Packet Loss Concealment

Packet Loss Concealment (PLC) is used at the receiver to mitigate or mask the effects of lost packets. Simple insertion schemes replace lost packets with noise, silence or with a copy of the previous packet. More complicated interpolation schemes use waveform substitution, pitch waveform replication or time scale modification.

Hybrid coders such as G.729 and G.723.1 have their own PLC algorithms as part of their codec specifications [24]. G.729 takes advantage of existing speech model and excitation parameters to generate a replacement speech segment. The synthesis filter parameters from the last good frame is used and a replacement excitation is synthesized to produce an approximation of the lost packet. Replacement packets signal levels are attenuated over time in the case of consecutive (burst) packet losses.

Waveform based coders such as G.711 also specify packet loss concealment algorithms as extensions to the standard [25]. Since waveform coders have no speech production model associated with the coder, regeneration techniques are more expensive because they require the estimation of the speech spectral parameters from scratch [26].

An additional problem caused by packet loss is that of state synchronization between source and receiver for state based codecs such as G.729 and G.723.1. A loss of state synchronization will cause errors for packets that arrive after one or more packets have been lost. Studies have shown that it takes up to 30 frames (300ms) for G.729 to fully resynchronize state, although audible degradation does not last nearly as long [27]. The effects of packet loss on state synchronization can be mitigated by using packets that arrive too late for playout (thus considered lost from a playout standpoint) to retroactively update or correct coder state [28].

2.3.5 Forward Error Correction

Forward Error Correction (FEC) is a source based mechanism aimed at compensating for the effects of packet loss by sending redundant information that aids the receiver in reconstructing the speech data in a lost packet or packets. *Media Independent* approaches make use of block codes such as a simple parity scheme or more robust Reed-Solomon

codes [24]. Other *media specific* approaches send a second copy of speech data contained in packet i along with the payload in packet $i + 1$, usually employing a more compact audio representation for the redundant data. This adds complexity as well as increasing the bitrate since the source must encode each speech frame twice using different codecs (or else double the bandwidth usage). It also requires that the receiver support multiple codecs.

FEC can add some robustness in the face of random packet loss, but is less resilient when faced with bursts of lost packets. While a simple FEC scheme may require little extra complexity, the increased bitrate required for more complex schemes make them suitable only to situations where bandwidth is not scarce or when FEC is used selectively to protect more perceptually important bits and/or packets [29].

2.3.6 Echo Cancellation

Echo is the term used to describe instances in which a speaker hears a delayed version of his own speech. When the echo delay path is short, echo is beneficial (sidetone). However, when the echo delay path exceeds 30ms it starts to become annoying to the user, and is considerably annoying for delays in excess of 50ms [30]. In these cases, echo cancellation should be performed by estimating and removing the echo from the signal.

There are two main sources of echo: hybrid echo and acoustic echo. The former occurs because of impedance mismatches. Electrical echo occurs in the PSTN, mostly at the juncture of the 2 wire local loop and the 4 wire trunk [2]. Electrical echo is usually only of concern to VoIP applications at connections between the PSTN and the IP network and is typically removed by echo cancellation performed at the gateway between the two networks [31].

Acoustic, or "multi-path" echo results from poor isolation of mouthpiece (or send path) from the earpiece (or return path) [32]. Acoustic echo can be attributable to lower quality headsets or speaker phones. In these cases, VoIP endpoints need to provide some form of echo cancellation.

2.3.7 Clock Skew

In many cases the actual rate of the source and receiver sampling clocks may differ despite running at the same nominal rate (i.e., 8 kHz). PC audio cards have shown to deviate

from their nominal rate by as much as ±0.5% [33]. Since all network delay information is garnered from RTP timestamps based on the sampling clocks, any difference between the actual sampling rates of source and receiver will appear as a gradual increase or decrease in calculated network delay. For simple playout scheduling algorithms, this can lead to an overflow or underflow of jitter buffers and may require explicit skew detection and compensation algorithms [34]. For many playout schedulers, however, the effect of skew can be neglected (see Section 2.4.4).

In an RTP environment, the RTP timestamp comes from a counter tied to a sampling clock. For two endpoints with the same nominal sampling clock rate, there is an offset between these sampling clock counters. Clock *skew* between two endpoints is used to refer to the rate of change of the offset between the two sampling clock counters. Clock *drift* is defined as the second derivative of the offset between sampling clocks, or alternatively, the rate of change of the clock skew. While clock drift is not assumed to be zero, it is assumed to average out to zero over time and be very small as compared to variations in network delays, and is thus considered negligible in any calculations [35].

2.4 Intrastream Synchronization

The goal of any intrastream synchronization algorithm is to preserve the temporal relationship of speech packets when presented for playout at the receiver [36]. This ordered and continuous reconstruction of the voice stream is subject to constraints on packet loss and/or end-to-end delay. The playout scheduling problem is illustrated in Fig. 2.4.

Intrastream synchronization, otherwise known as *playout scheduling*, has been an area of active study since the beginnings of packet voice networks in the 1970's. The need for "an ordered synchronous sample flow in the face of stochastic network behaviour" [37] requires a "delay vs. rhythm" tradeoff at the endpoint [38].

While some algorithms simply adjust the buffering delay of received packets in order to maintain a safe level of buffer occupancy [36, 39], most rely on timing information in the form of timestamps. Early attempts at defining a protocol [40] for real-time applications over packet networks eventually led to the adoption of RTP as a standard in 1996 [8]. As discussed in Section 2.2, RTP provides information on packet ordering and the *relative* delay, given in units of *sampling instants*, of packets in a given stream. While the timestamp itself does not provide any information on the *actual* network delay incurred by a packet,

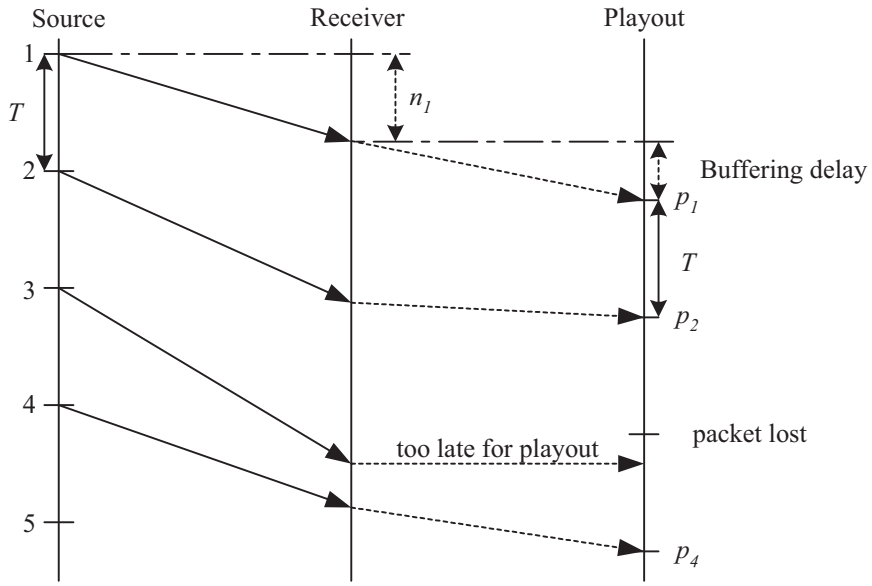


Fig. 2.4 Illustration of Playout scheduling problem. Received packets are buffered such that they can be be played out at intervals of one packet (T). Packet 3 arrives too late for playout and is thus considered lost.

it provides sufficient information to track network delay variation over the lifetime of an audio stream (neglecting clock skew).

2.4.1 Fixed Playout

The *playout delay* of a packet can be defined as the time between a packet leaving the transmitter and being played out at the receiver. This includes network delay as well as buffering delay. A simple playout algorithm will use a *fixed* value for playout delay over all packets for the duration of an audio session [41, 42]. Given an acceptable packet loss rate l, the playout delay, $\tilde{d}_l$, is chosen such that:

$$\forall i \in P[n_i < \tilde{d}_l] = 1 - l, \tag{2.2}$$

where n_i is the network delay of the ith packet. The first packet to be played out will have a playout time p_1 calculated as:

$$p_1 = a_1 + \tilde{d}_l - n_1, \tag{2.3}$$

where a_i is the arrival time of packet i, $\tilde{d}_l$ is the playout delay, as calculated in Eq. (2.2), and n_i is the network delay of the ith packet (calculated from the packet RTP timestamp). To ensure continuous playout, all subsequent packets must be played out in a periodic manner and thus have playout times calculated as:

$$p_i = p_{i-1} + T, \tag{2.4}$$

where T is the length of the packet. One problem with the fixed playout approach is that it requires prior knowledge of the delay characteristics of the network (usually achieved through a warm-up period). Another more significant problem is that it is unable to react to changes in those network delay characteristics, or adapt to skew between source and destination sampling clocks [41].

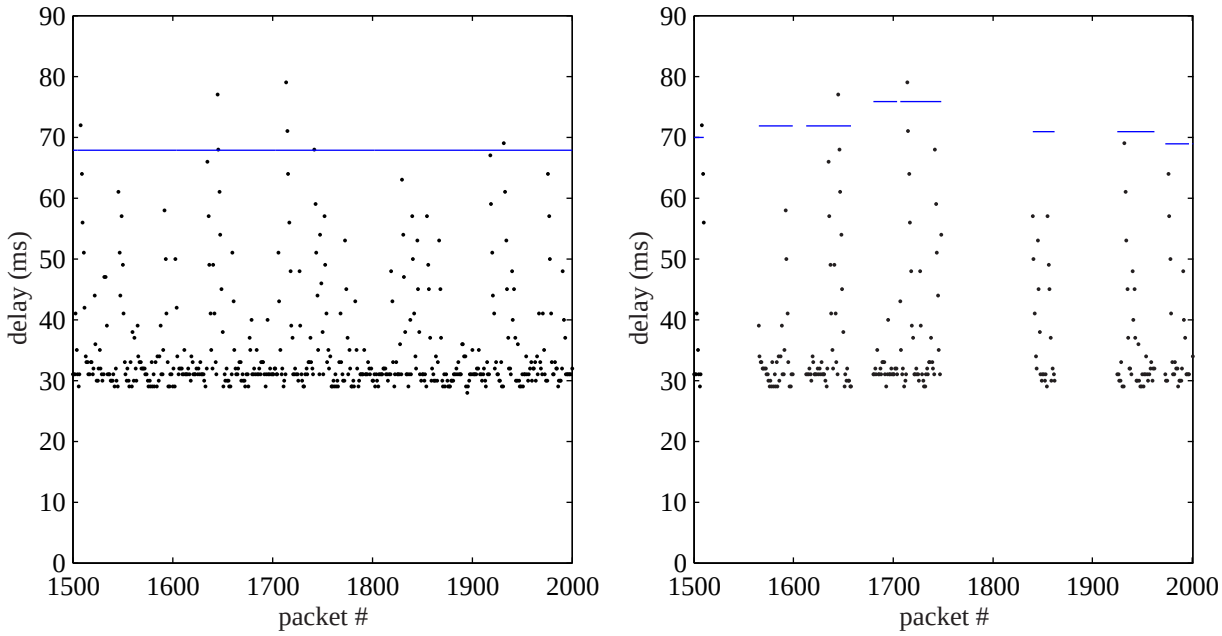
2.4.2 Talkspurt Adaptive Playout

Talkspurt adaptive playout scheduling algorithms [41–45] allow for a readjusting of the playout delay at the start of every talkspurt. By shortening or lengthening the silence periods between talkspurts, the playout delay can be changed without disturbing the temporal relationships of packets within talkspurts. Artificial compression or stretching of silence periods between talkspurts by small amounts does not affect the perceived quality of speech [43]. The ability to modify the playout delay at the beginning of each talkspurt allows for adaptation to changes in network delay distributions.

One of the first playout scheduling algorithms to be explicitly defined and published was that of Ramjee et. al. in 1994 [41]. At the beginning of each talkspurt, the playout time, p_i, of packet i is calculated as:

$$p_i = t_i + \hat{d}_i + \beta \hat{v}_i, \tag{2.5}$$

where t_i is the time at which packet i is generated at the source host, $\hat{d}_i$ and $\hat{v}_i$ are estimates of the network delay mean and variation during the talkspurt, and β is a *safety factor* (chosen as 4 in [41]) related to the acceptable packet loss rate. Any subsequent packets arriving in the same talkspurt will be subject to the same playout delay as the first packet,



(a) Fixed Playout Scheduler

(b) Talkspurt Adaptive Playout Scheduler

Fig. 2.5 Playout Schedulers. Dots represent the network delay of each packet, while the solid line represents the end-to-end delay (network + buffering). Dots above the line represent packets that arrived late for their scheduled playout and are thus considered lost. In (b), gaps where there are no dots represent silence periods.

with playout time calculated as in the fixed playout case:

$$p_i = p_{i-1} + T \tag{2.6}$$

Ramjee proposes four methods for estimating $\hat{d}_i$ and $\hat{v}_i$. The first (Algorithm 1) uses an Autoregressive (AR) estimate. The delay mean estimate is computed as:

$$\hat{d}_i = \alpha\ \hat{d}_{i-1} + (1-\alpha)\ n_i, \tag{2.7}$$

and the variation estimate is computed as:

$$\hat{v}_i = \alpha \hat{v}_{i-1} + (1-\alpha)|\hat{d}_i - n_i|, \tag{2.8}$$

where n_i is the network delay of the ith packet as calculated from the packet timestamp, and α is a weighting factor which determines how quickly the algorithm will adapt to changes in network conditions. In [41], α was chosen to be 0.998002.

Ramjee's Algorithm 2 is the same as Algorithm 1, except that a value of 0.75 is used for the weighting factor, α, when the network delay n_i of the current packet is greater than the current delay estimate $\hat{d}_i$. This decreases the chances of packet loss by allowing the delay estimate to adapt more quickly in times of increasing network delay.

Algorithm 3 is based on the algorithm used in NeVoT (Network Voice Terminal) 1.6 and uses the lowest network delay value in the previous talkspurt as the delay estimate $\hat{d}_i$ [46]:

$$\hat{d}_i = \min_{j \in S_i} \{n_j\} \tag{2.9}$$

where S_i is the set of all packet delays received in the previous talkspurt.

Ramjee's Algorithm 4 introduces a *spike detection* component. Studies of IP network delay traces show frequent occurences of *delay spikes*, conjectured to arise due to the accumulation of packets in a router queue [47, 48]. Spikes are characterized by a large increase in network delay, followed by the arrival of many packets in quick succession. In algorithm 4, a spike mode is triggered if there is a sudden increase in network delay between successive packets. Once in spike mode, the estimate of the delay mean, $\hat{d}_i$, is dictated only by the most recently observed delay value so as to adapt as quickly as possible to the spike:

$$\hat{d}_i = \hat{d}_{i-1} + n_i - n_{i-1} \tag{2.10}$$

Spike mode is exited when the difference between successive packet delays becomes less than a given threshold, and normal operation resumes until the next spike is detected [41].

While an AR estimate based method was used in [41] to estimate the network delay mean and variation and determine a suitable playout delay , other approaches use histogram based estimates [44, 45, 49], or adaptive filter based estimates [50, 51] to calculate the appropriate playout delay. In the histogram approach, the past N delay values are stored in an ordered array. Given a loss rate, l, the stored delay value for which $l\%$ of delays are higher is chosen as the playout delay for the given talkspurt. The playout delay is thus chosen via a statistical approximation of the l^{th}-percentile network delay. As new packets arrive, the oldest stored delay value is removed from the ordered array, and the current

delay value is added. Values used for N can range from 35 in [52] to 10000 in [44]. A large N has the advantage of an accurate representation of network delays over a sample size that is likely to include all network variations. The actual packet loss rate experienced by a given choice of playout delay will be very close to the target loss rate l. Disadvantages of a large N are susceptibility to clock skew between source and receiver (see Section 2.4.4), larger memory requirements, and the inability to adapt to quickly changing network delays. A small N allows for quick adaptations to network delays (and clock skew) but results in a larger discrepancy between the target loss rate l and the actual packet loss rate experienced for a given choice of playout delay. Histogram based algorithms generally incorporate some form of spike detection algorithm, similar to the one used in [41].

Adaptive filter based algorithms [50, 51] attempt to predict the network delay of the ensuing packet by minimizing the expected mean square error between the actual data and the estimate, using the Normalized Least Mean Square (NLMS) algorithm [50]. The estimate for the network delay d_i is given by:

$$\hat{d}_i = \boldsymbol{w}_i^T \boldsymbol{n}_i \tag{2.11}$$

where $\boldsymbol{w}_i$ is the $N \times 1$ vector of adaptive filter coefficients and $\boldsymbol{n}_i$ is the $N \times 1$ vector containing the last N network delay values. The filter coefficients are updated after the reception of a new packet using the NLMS algorithm [50]:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \frac{\mu}{\boldsymbol{n}_i^T \boldsymbol{n}_i + a} \boldsymbol{n}_i e_i, \tag{2.12}$$

where μ is the step size, a is a small constant to prevent division by zero, e_i is the estimation error ($e_i = \hat{d}_i - n_i$). Adaptive filter based algorithms generally track network delay variations much more closely than histogram or AR based methods, and are thus best suited to per-packet adaptive playout algorithms.

2.4.3 Packet Adaptive Playout

A second and more recent class of adaptive playout scheduling algorithms allow for the modification of the playout delay on a per-packet basis [52–55]. This is achieved by scaling individual packets in time using Waveform Synchronized Overlap-Add (WSOLA) [52] or related methods [53–55].

Liang introduces a per-packet adaptive playout scheduling algorithm in [52]. He uses a histogram based method with an N value of 35 to determine the target playout delay. A new target playout delay, parameterized by a target loss rate l, is calculated upon arrival of each packet. If this playout delay differs from the previous calculation of the target playout delay, the packet is scaled by the difference between the two playout delay estimates. If d_i is the target playout delay as calculated after reception of packet i and d_{i+1} is the target playout delay as calculated after reception of packet $i+1$, then packet $i+1$ will be scaled to length L_{i+1} calculated as:

$$L_{i+1} = d_{i+1} - d_i + T, \tag{2.13}$$

where T is the original packet length of packet $i+1$. The playout time, p_{i+1}, of packet $i+1$ is dictated by the playout time of packet i, as well as its length:

$$p_{i+1} = p_i + L_i. \tag{2.14}$$

In general, the target length of a scaled packet cannot always be precisely achieved. The WSOLA operation works on integer multiples of pitch periods, making it not always possible to scale packets to arbitrary lengths [56]. The actual scaled packet length will often differ from the target packet length. To avoid frivolous time scaling operations, a minimum scaling threshold is introduced, and any packet targeted to be scaled by an amount less than this threshold is left at its original length. In addition, a maximum scaling threshold is introduced such that packets are not scaled by an amount that would introduce perceptible degradation in the quality of the scaled speech signal [52].

While Liang uses a histogram based approach in [52] to determine the target playout delay, other approaches in determining target playout delays, such as autoregressive estimates or adaptive filter based algorithms, are also valid and can be incorporated into a per-packet adaptive playout algorithm.

2.4.4 Effect of Clock Skew

As discussed in Section 2.3.7, source and destination sampling clocks may differ in frequency by as much as 0.5% [33]. The perceived effect of clock skew at the receiver is that of a slow increase or decrease in the network delay. If the source sampling clock is faster than that

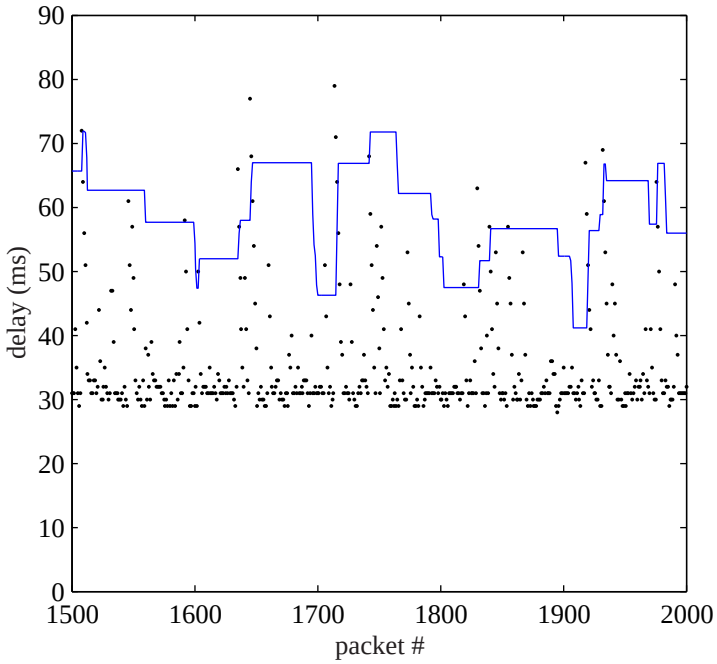
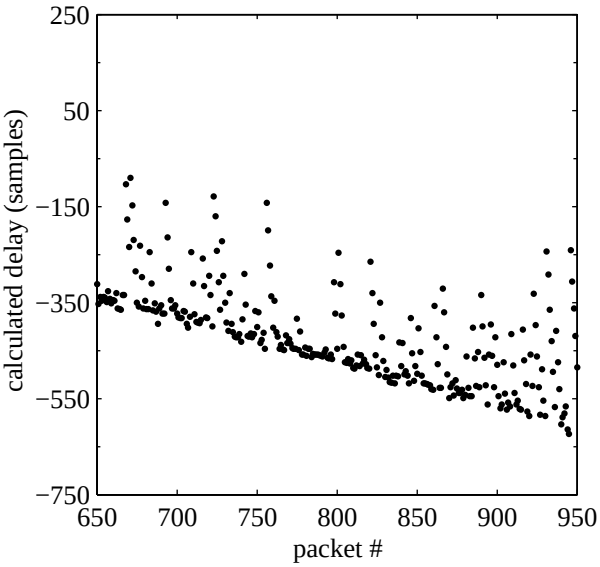


Fig. 2.6 Packet Adaptive Playout Scheduler.

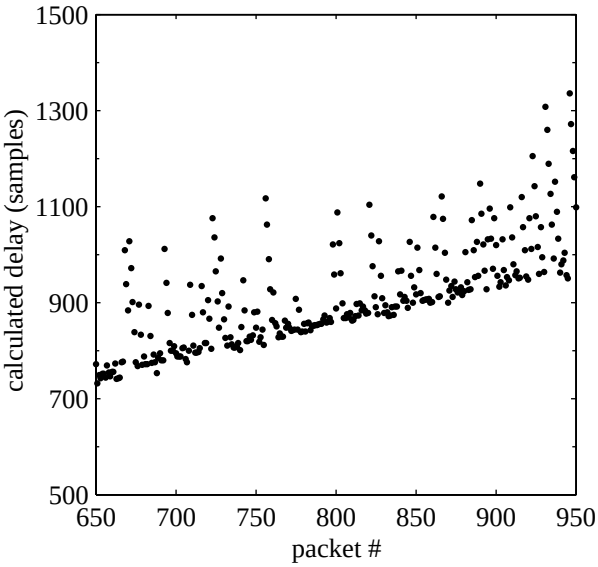
of the receiver (positive skew), the receiver, based on RTP timestamps, will see a *perceived* decrease (on average) in network delay. The opposite occurs if the receiver sample clock is faster than that of the source (negative skew).

Playout scheduling algorithms most susceptible to clock skew are those that use network delay values from packets received far in the past in order to calculate the playout delay, or equivalently, those that react slowly to changes in network characteristics. In these cases, the error introduced by the clock skew in the playout delay estimate is significant as compared to the variability in network delay. Algorithms such as in [44], using histogram based estimates with $N = 10000$, are especially susceptible to clock skew. Assuming 20ms packets, 0.5% clock skew would lead to an offset error of 1 s in the calculated delays between the first and last packets in the histogram. For these slowly adapting playout schedulers, clock skew needs to be handled explicitly by using an additional clock skew compensation mechanism, such as that proposed in [34].

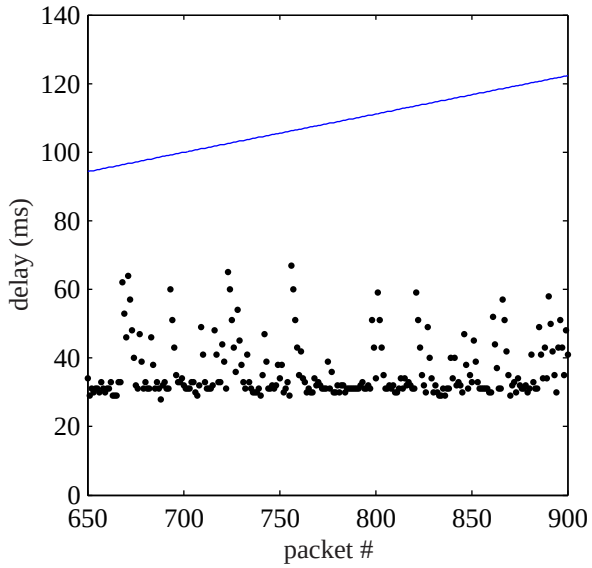
Algorithms that adapt quickly to network changes, such as in [52], are robust against clock skew because only the past 35 delay values are used to form the playout delay estimate. The error introduced by the clock skew becomes negligible as compared to the variability



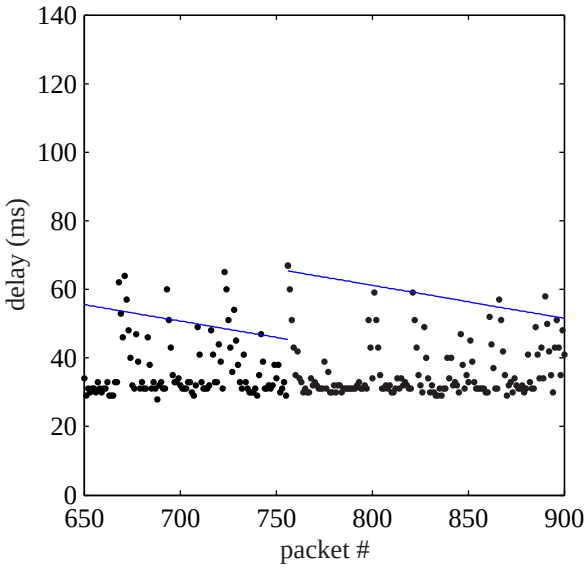
(a) Delay as calculated from RTP timestamps in face of positive clock skew.



(b) Delay as calculated from RTP timestamps in face of negative clock skew.



(c) Playout Scheduling in the face of positive clock skew.



(d) Playout Scheduling in the face of negative clock skew.

Fig. 2.7 Effect of Clock Skew. A sample of the effects of clock skew on playout scheduling. Here, the playout scheduling algorithm is fixed and histogram based with a histogram length, N, of 300. Positive clock skew results in excessive buffering delays, while negative clock skew results in excessive packet loss.

of the network delays.

Endpoints participating in a conference may be susceptible to clock skew regardless of the playout algorithm used. Conferencing architectures in which a centralized bridge selects and forwards only a subset of the packets in a given stream (see Section 2.7.1) can create scenarios in which one conference participant has not received any packets from another particular conferee for a substantial period of time because that conferee has not been selected as a speaker. When that conferee is reselected as a speaker, other conference endpoints will have to make playout scheduling decisions based on packets received far in the past (from the last time that conferee had been selected as a speaker).

2.4.5 Playout Scheduling Performance Metrics

In general, playout algorithms are parametrized by a tolerable or target rate of late packets (target loss rate), whether explicitly [44, 52] or more loosely by some form of safety factor, as in [41]. Some newer algorithms attempt to use the combined effects of end-to-end delay and packet loss so as to optimize the perceived quality of the output audio [57, 58] while others also consider the effects of Forward Error Correction (FEC) in their optimizations [59, 60]. As RTP timestamps only provide a value for the relative network delay on received packets, external mechanisms, such as RTCP reports, NTP, or the use of probe packets need to be used in order to estimate the actual end-to-end delay. The effects of packet loss and end-to-end delay on speech quality are generally combined by way of the E-model (ITU-T G.107) [57, 60, 61] or some variant thereof [58]. The E-model, given multiple transmission impairment parameters, including packet loss and end-to-end delay, will output a scalar quality rating, R, measuring perceived conversation quality, which in turn can be mapped into an MOS score. The E-model, however, is not meant for small scale optimizations, but more as a guide for the planning of large scale networks. The traditional metric for evaluating the performance of a given playout algorithm is an end-to-end delay vs. packet loss graph for a given set of network conditions.[1]

[1] While the E-model and other [58] methods can attempt to give an optimized "perceptual" evaluation of the combined effects of loss and delay for a given set of network conditions, the end-to-end delay vs. packet loss graph will be used in this thesis so as to remove any debate about the validity of the underlying model combining their perceived effects.

2.5 Interstream Synchronization

While intrastream synchronization (playout scheduling) attempts to preserve temporal relationships between packets in a given stream, *interstream* synchronization attempts to preserve temporal relationships between two or more different media streams. Different applications have different levels of required synchronization. Some streams are very tightly coupled (stereo audio), while others have less strict synchronization requirements (lip-synch of video and speech) and others still will have a very loose temporal relationship (background music and corresponding video) [62]. The problem of interstream synchronization is simplified if the streams originate at the same source, but in many cases temporally related media will originate from multiple distributed sources. For voice conferencing applications, it has been shown that *synchronization errors* less than ±120 ms are generally not noticeable [62].

The synchronization error between two streams can be defined as the difference in generation times of streams being played out at the same time [63], or alternitavely as the difference in end-to-end delay experienced by packets played out at the same time. Akyildiz identifies four sources of synchronization errors, shown in Fig. 2.8 [64]:

- Clock skew
- Different initial collection times
- Different initial playback times
- Network delay jitter

Clock Skew

Clock skew between two sources of temporally related media will cause a gradual increase in synchronization error as one stream will be played out faster than the other at the receiver. As in the case of playout scheduling (see Section 2.4.4), clock skew is in general negligible as compared to the network delay jitter experienced in wide-area packet networks.

Different Initial Collection Times

Distributed sources of temporally related media may begin transmitting data at different times and thus have unsynchronized packet boundaries. Unless the streams to be synchro-

nized come from the same source, synchronizing initial collection times among distributed sources requires clock synchronization between each source and control messages between all sources so as to begin transmission at the same time [65, 66].

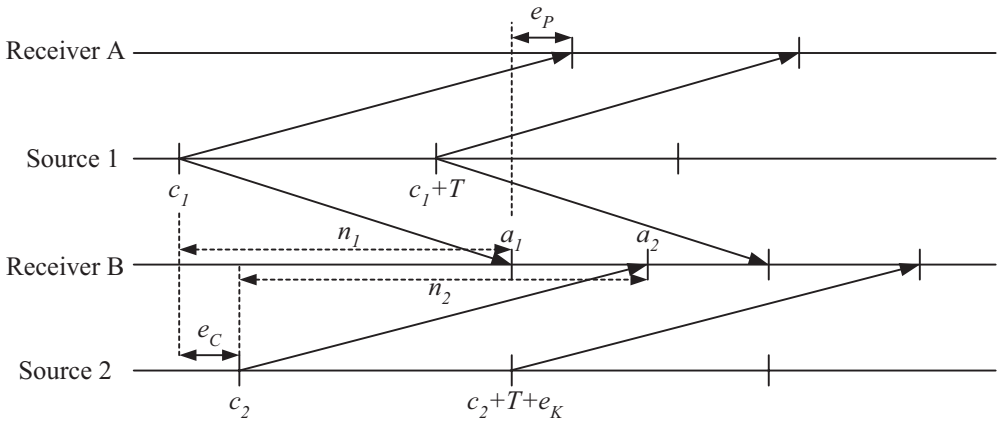


Fig. 2.8 Interstream Synchronization Errors. Here, e_c is the error introduced by different initial collection times, e_p is the error introduced by different initial playback times, e_k is the error introduced by clock skew. The error introduced by network delay jitter is $n_1 - n_2$.

Different Initial Playback Times

When there are multiple distributed receivers for which a given packet is destined, each source to destination path will experience different network delays. This results in some receivers obtaining the packet before others (see Fig. 2.8). In a conferencing scenario, this can result in different conferees hearing the same audio at different times.

The problem of different initial playback times is often referred to as *group synchronization* and can be considered as a complimentary problem to interstream synchronization [67]. Group synchronization is generally only required for applications with a need for *fairness*, in which a set of distributed participants should all receive the same media at the same time. Networked video games and online auctions are two such scenarios [68, 69].

Group synchronization is generally achieved through feedback mechanisms between destinations detailing the worst case delay as experienced by all participants [67]. All participants then inject additional delay through buffering in order to experience this worst case delay and thus maintain a sense of *fairness* among the group.

Network Delay Jitter

Temporally related streams experiencing different network delays and delay variations is the main source of synchronization errors, especially when those streams come from distributed sources. In Fig. 2.8, differences in delay for packets travelling from Source 1 and Source 2 result in synchronization errors at Receiver B.

Most approaches to mitigate synchronization errors caused by network delay jitter use some form of synchronized clock between distributed media sources [66, 69], although some techniques attempt synchronization relying only on packet timestamps [63, 70].

Interstream synchronization algorithms can be classified into one of three categories: (1) those that use globally synchronized clocks as well as control messaging, (2) those that use globally synchronized clocks but no control messaging, and (3) those that rely solely on timestamps from packets in the actual data stream.

2.5.1 Timestamp-based Algorithms with no Global Clock

Algorithms that do not have a global clock rely on timestamps and packet arrival times to synchronize playout of packets from different streams. If the timestamps of the streams to be synchronized are not based on the same clock (i.e., if the streams come from different sources), timestamps will only provide a relative delay for packets in a given stream.

In [70], Rangan shows that if the network delay jitter on any of the streams to be synchronized is greater than the length of one packet, then it is not possible to reliably map packets from one stream to another without synchronized clocks.

Algorithms that only use timestamps for synchronizing streams are best suited to applications where the streams to be synchronized come from the same source [63]. In these cases, timestamps from the streams to be synchronized will have been generated with the same clock, and thus be effectively synchronized.

While accurate synchronization of streams arriving from distributed sources is generally not possible using only timestamps, such approaches can still be useful for mapping packet boundaries for the purposes of mixing. They can also provide adequate synchronization in cases where the network delay variations between streams are below a perceptible synchronization error threshold (for the application in question).

2.5.2 Timestamp-based Algorithms with Global Clock

Global clocks can be established through the use of NTP, which allows for a mapping between a local sampling clock and a global time [65]. This mapping gives the receiver an absolute delay experienced by a given packet, and the means to calculate that packet's generation time. Algorithms presented in [63, 65, 71] follow a similar approach, which is summarized in the ensuing paragraphs.

The collection time (or transmission time), c_j, of a packet from stream j can be calculated as:

$$c_j = a_j - n_j, \tag{2.15}$$

where a_j is the arrival time of the packet and n_j is the absolute delay calculated from timestamps and the use of a global clock. The playout delay, d_j is then:

$$d_j = p_j - c_j, \tag{2.16}$$

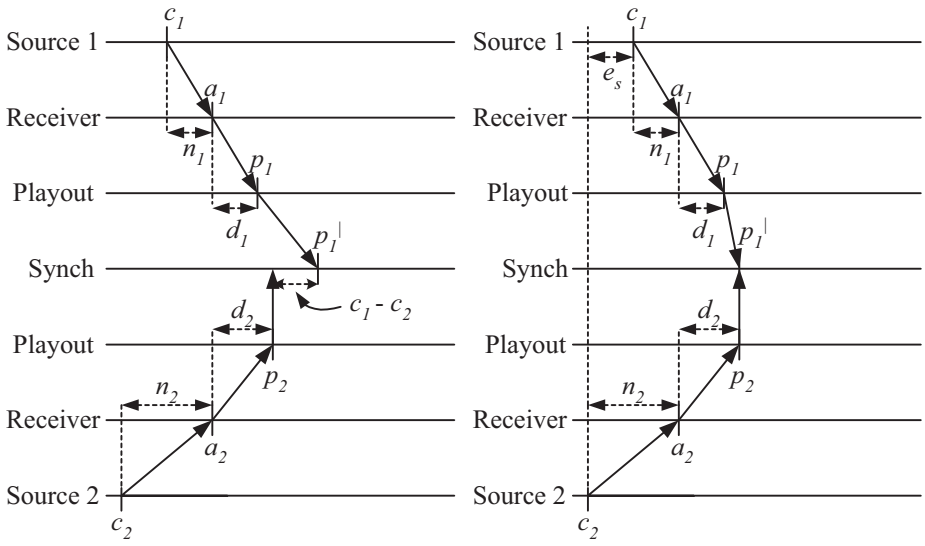
where p_j is the playout time as calculated via some playout scheduling algorithm. If we consider streams 1 and 2, where d_2 is greater than d_1, then the playout time of stream 1, p_1 , is adjusted such that the end-to-end delay experienced by each stream is the same:

$$p_1' = p_2 + (c_1 - c_2). \tag{2.17}$$

The synchronization error, e_s, can be calculated by taking the difference in playout delays between the two streams, $d_2 - d_1$. When synchronization is performed as described above, the error is zero:

$$\begin{aligned} e_s &= d_2 - d_1 \\ &= (p_2 - c_2) - (p_1' - c_1) \\ &= (p_2 - c_2) - ((p_2 + c_1 - c_2) - c_1) \\ &= 0 \end{aligned} \tag{2.18}$$

In this case, stream 2 is referred to as the *master* stream, and stream 1 is referred to as the *slave* stream. In most algorithms, the master stream is the one exhibiting the greatest



(a) Mixing on a sample-by-sample basis.

(b) Mixing on a packet-by-packet basis.

Fig. 2.9 Interstream Synchronization using Global Clock. In (a), the synchronization error is zero. When mixing on a packet-by-packet basis as in (b), their will be a synchronization error, e_s, equal to the difference in initial collection times of the streams to be synchronized.

network delay [65, 71]. When more than 2 streams are to be synchronized, all other streams are individually synchronized to the master stream in the same way as the playout of stream 1 was adjusted in the preceding case.

In some cases, packets will need to be mixed on a packet-by-packet basis (as opposed to on a sample-by-sample basis). In these cases playout times can not be adjusted by an arbitrary number of samples in the synchronization stage, as in Eq. (2.17), and must be done in such a way that packet boundaries line up:

$$p_1' = p_2 \pm \ mT, \tag{2.19}$$

where T is the packet length, and m is chosen so as to minimize the synchronization error. If the packets in question were generated less than half of one packet period apart, then m will be zero and p_1' will equal p_2. The synchronization error in this case will be dependent on the difference in initial collection times of the two streams:

$$\begin{aligned} e_s &= d_1 - d_2 \\ &= (p_1' - c1) - (p_2 - c2) \\ &= (p_2 - c_1) - (p_2 - c2) \\ &= c_2 - c_1 \end{aligned} \tag{2.20}$$

A synchronization error of zero when mixing on a packet-by-packet basis can only be achieved if the initial collection times are the same or a multiple of a packet period apart (and thus synchronized). Sample-by-sample and packet-by-packet mixing scenarios are illustrated in Fig. 2.9.

If the master stream changes, other streams are adjusted as above, but this time based on the playout delay of the new master stream. In general, *intrastream* synchronization will take precedence over interstream synchronization [71], and interstream synchronization to the new master stream may not be possible until such a time where the temporal relationship of packets within a given stream can be changed[2].

2.5.3 Timestamp-based Algorithms with Global Clock and Control Messaging

Having a global clock can eliminate the synchronization error introduced by network jitter, but some form of control messaging is required to synchronize initial collection times when mixing of synchronized streams is to be done on a packet-by-packet basis. Control messaging is also required to perform group synchronization [64, 66, 67].

Rothermel proposes a *Start-up* protocol in [66] in order to ensure that all sources begin transmission at the same time. A controller sends a specified common start-up time to all sources in a synchronization group. This requires that all sources have synchronized clocks. Similar approaches for synchronizing start-up times are used in [64] and [65].

Ishibashi extends the concept of interstream synchronization to group synchronization in [67]. The master stream is common to *all* receivers, and is chosen as the stream for which the delay is longest (over all streams at all receivers). Control messages are required to broadcast the worst-case (master) playout delay to all other receivers. Additional control messaging is required any time the master stream changes. A similar technique is used

[2]Intrastream adjustments generally only occur during silence periods.

in [66].

2.6 Centralized Conferencing

The most prevalent form of conferencing is of a centralized nature, in which conferees dial into a bridge. The role of a conventional bridge is to mimic the rest of the conference to a given conferee by summing the contributions of other conferees and returning a composite signal. The bridge is responsible for abstracting conference endpoints from the connection and call processing details involved in conference setup. From an endpoint's perspective, it is like having a one on one conversation with the bridge. The bridge achieves this abstraction by selecting a portion of conferees incoming voice streams as *speakers*, and constructing a composite signal from those selected speakers. The composite signal sent to each conferee may be different, as it is important that any signal contribution from a given conferee not be returned to that same conferee [1].

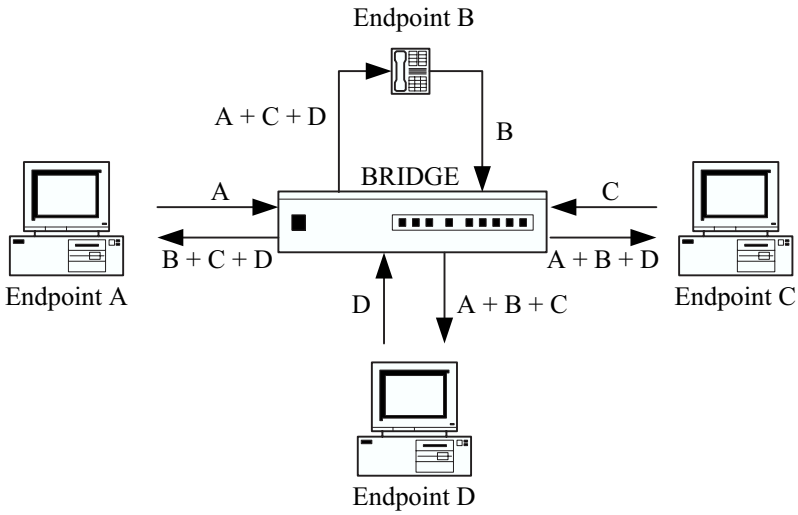


Fig. 2.10 Generalized Centralized Conference. In general, the bridge will only select a certain number (usually 2 or 3) of speakers from which the composite signal is created.

Conventional bridges differ in the number of designated speakers, M, the algorithm used to select those speakers, and the way in which they construct the composite signal made up of those speakers.

An additional challenge for conferencing in best-effort packet networks such as the

Internet is the variability in delay of incoming packets. Not only do bridges have to deal with intrastream synchronization to smooth out delay jitter within a given stream, but they also have to worry about the difference in delay and delay variations of streams originating from different conferees.

2.6.1 Speaker Selection

In general, a conference made up of N participants will have a limit, M, on the number of speakers that will be active at a given time. If the composite signal is made up of all N conferees signals (i.e., $M = N$), the accumulation of background noise from the non-speaking participants will degrade the overall quality of the composite signal [2, 72]. A choice of M that is less than N will reduce background noise as well as complexity in the mixing and synchronization operations.

While conversational dynamics of two-way conversations are fairly well modelled [73], less is known about conversation dynamics of larger conferences. Studies in [74, 75] show that two or more people talk simultaneously for only 6–11% of the conference time, on average. The nature and size of a conference may vary considerably and will dictate the interactivity and frequency of interruptions [74], but it is shown in [76] that for an active conference (i.e., back and forth with many interruptions) of four people, rarely were there more than two people talking at the same time. In general, a conference can be well represented by selecting only two active speakers at any given time.

Speaker selection algorithms aim to choose M active speakers from N conferees. This selection process is done based on features of the incoming voice streams, such as speech energy and VAD decisions. Time-based criterion, such as a first come first served type approach, are also used, to prevent spurious switching between conferees [5].

Smith presents a Multi-Speaker/Interrupter(MS/I) algorithm in [76] which combines the use of time-based criterion and level-based criterion, while allowing for speaker interrupts. The algorithm ranks speakers based on each conferee's *power signal envelope*, $\hat{E}_i$, which is updated with the arrival of each packet:

$$\hat{E}_{i+1} = \max(\bar{E}_i, \beta\hat{E}_i + (1-\beta)\bar{E}_i), \tag{2.21}$$

where $\bar{E}_i$ is the signal power of the current packet, and β is the weight of the exponential average. It is assumed in [76] that the signal power is carried as side information on

upstream packets, although alternatively it could be calculated at the bridge.

A barge-in threshold, β_{th} is used to avoid spurious switching between participants. In order to pre-empt a currently selected speaker, a new speaker must surpass the power signal envelope of the incumbent talker by β_{th}. A conferee of priority m will be selected over a conferee of priority $m-k$ only in the event that

$$10\log(\frac{\hat{E}_i^{(m)}}{\hat{E}_i^{(m-k)}}) > \beta_{th}. \tag{2.22}$$

2.6.2 Signal Equalization

Conferees may have different incoming signal levels, due to transmission facilities, distance from mouth to speaker, or simply natural voice level [76]. Since speaker selection algorithms use speech energy as a determining characteristic, it is important to equalize speech levels over all conferees. Automatic Gain Control (AGC) is used to minimize differences in average speech level by scaling each incoming signal towards a target level. Scaling of voice streams can also be used to accentuate the contributions of a given speaker [77].

2.6.3 Mixing

Because the bridge must ensure that contributions from a given conferee are not sent back to that same conferee (this will cause an annoying echo effect), the bridge must generate a different composite signal for each selected speaker. If $M = N$ (i.e. all conferee contributions make up conference sum) then each conferee will receive a unique signal summation comprised of the other $N-1$ signals. In general, for $M < N$, there will be $M+1$ different composite signals.

Mixing is typically done by the bridge in one of two ways: by creating unique sums for each of the $M+1$ signals, or by creating one composite signal and subtracting the speaker in question's contribution for the composite signal destined for that speaker in question. The first approach will take M^2 additions, while the second will take $2M$ additions. Both approaches are equivalent in complexity for $M = 2$.

The actual mixing of streams is done by simple addition of linear PCM samples. Incoming packets must be decoded into linear PCM, added to form the required composite signals, and then re-encoded. This tandem encoding will degrade the speech quality, especially for lower bit rate coders such as G.729 or G.723.1 (see Section 2.3.1). If conference

endpoints are using different codecs, the bridge must perform the re-encoding step once for each different codec.

An alternative approach to the bridge mixing multiple streams is to have one composite mix made up of all M speakers, and have endpoints subtract out their own contribution from the mixed packet upon reception from the bridge. This approach would require additional synchronization information in the RTP header of the mixed packet in order to allow endpoints to properly line up the samples to be subtracted, as well as increased functionality in the endpoint in order to perform this echo cancellation procedure. Additionally, because of the decode-mix-encode operation at the bridge, the original samples (to be subtracted from mixed packet) may differ from the contributing samples in the mixed packet, thus adding distortion when subtracting the original packet from the mixed packet. This is especially true for non-waveform codecs (i.e. G.729 and G.723.1). One advantage of this approach is that the bridge will send the same composite signal to all conferees for the entirety of the conference, eliminating any need for the bridge to have separate codec state (and thus perform a separate encoding procedure) for each endpoint stream (see Section 2.6.5).

2.6.4 IP conferencing

Performing centralized mixing in an IP environment is complicated by the relatively high network delay jitter within a given stream as well as differences in delay and delay jitter between different streams. Intrastream synchronization is required to smooth delay jitter on a given stream, while some form of interstream synchronization mechanism is required to map packets from one stream to the appropriate packet in another stream, if they are to be mixed.

Simard et. al. present a design for packet-based conferencing in [3]. The design allows for the sending of a lone talker or alternatively a composite signal made up of two selected talkers when in periods of multi-talk. In periods of single-talk, the lone talker is immediately encapsulated and sent to all other conference participants. In periods of multi-talk, the bridge, upon reception of a packet from the *primary* speaker, will wait upon for an amount of time, T, for the corresponding packet to arrive from the secondary speaker. If the secondary speaker's packet arrives within that time, T, the packets are decoded, mixed and re-encoded and the composite signal is sent to other non-speaking conferees. If a time,

T, elapses before the arrival of the secondary speaker's signal, a voice signal is *generated* for the secondary speaker, and the primary and generated secondary packets are mixed and forwarded[3] [3].

2.6.5 Codec Issues

State dependent codecs, such as G.729 and G.723.1 provide additional complications when used in a conferencing environment in which a centralized bridge performs mixing and/or speaker selection. The bridge must ensure that state synchronization is kept at the endpoints, which involves a logical separation of codec state in all bridge-to-endpoint streams.

As an example, consider a bridge that selects M speakers from an N person conference. As discussed in Section 2.6.3, the bridge will remove the signal contributions of an endpoint from the composite mix when sending packets to that particular endpoint. This means that over the course of a conference (provided all conferees become active speakers at some time), the total voice stream sent from the bridge to the endpoints will be different for each endpoint. Because there are N unique voice streams, the bridge needs to keep codec state for all N streams. In addition, although the bridge will only create $M + 1$ different mixes for a given packet (see Section 2.6.3), it will need to perform N different encodings, because each stream will have a different codec state.

Many codec state synchronization requirements can be foregone at the expense of a small degradation in speech, as state synchronization errors occur mostly between talkspurts, during speaker transitions. Codec state synchronization issues will be discussed in further detail in Section 3.3.6.

The use of low bit-rate codecs can also cause noticeable degradation in speech quality of the composite signal created at the bridge due to the tandem encodings caused by the encode-mix-decode process. In addition, a multi-talker signal is being encoded with a coder that has been optimized for a single talker. One approach to mitigate the effects of the tandeming problem is to adapt M in periods of multi-talk. When there is only one active talker in the conference, that talker's stream is selected as the composite signal and no decoding or mixing is required at the bridge. Tandeming is restricted to times of multi-talk, at which point the M active talkers are decoded, mixed and re-encoded as in the conventional case [13, 78]. This approach can lead to problems when using codecs

[3]The secondary speaker source will receive only the primary speaker's contribution and the primary speaker will receive only the secondary speaker's contribution.

employing a look-ahead, when switching from a period of single-talk to a period of multi-talk [13].

In order to fully eliminate any tandeming of coded speech, a tandem-free architecture is required, which involves offloading the decode-mix operation to endpoints.

2.7 Tandem-Free Conferencing

Tandem-free Conferencing seeks to avoid any tandem encodings that occur when a bridge must decode and mix multiple signals before re-encoding the signal sum. The tandem-free concept was initially proposed by Forgie in 1979 as a means of avoiding speech quality degradation due to tandem encodings of low bit-rate coded speech [74].

Tandem-free operation is achieved by performing only speaker selection at the bridge, forwarding M streams to each endpoint (or $M-1$ streams if that endpoint is one of the speakers), and leaving the endpoints to do the mixing of multiple streams. This approach requires additional support in the endpoint for mixing, as well as support for any codecs used by other conferees. The bridge can extract speech parameters used for speaker selection by doing a partial decode of incoming packets, although speaker selection is simplified if endpoints provide side information (speech energy and VAD decision) on upstream packets. In this way, the bridge need not decode any packets in order to perform speaker selection [76].

A tandem-free architecture is also well suited to secure conferencing. If the endpoints provide side information on upstream packets, then the speech payload can remain encrypted for the entire source endpoint to destination endpoint path. In this way, the conference bridge need not be a "trusted" entity.

2.7.1 Select-And-Forward Bridge

The *Select-And-Forward* bridge acts as a signal selector, forwarding packets selected as speakers, and dropping those that are deemed to be not speaking. All decode and mixing operation are offloaded to endpoints. The offloading of the mixing operation to endpoints eliminates tandem encoding of signals and can simplify synchronization operations at the bridge, as no interstream synchronization is required because voice streams are not combined. Intrastream synchronization can also be off-loaded to endpoints.

The concept of a Select-And-Forward bridge was originally proposed by Forgie in 1979. Work by Simard et. al. [79] and extended this concept to multiple speakers, while Smith provided a more detailed design in [5].

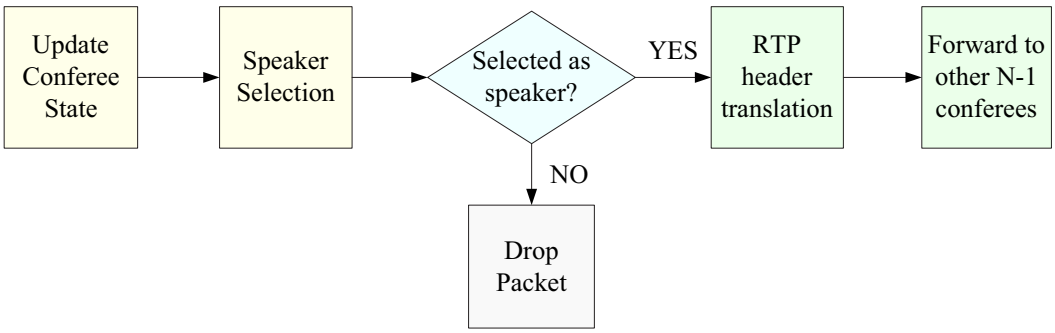


Fig. 2.11 Select-And-Forward Bridge Packet Reception. RTP header translation will be different for every outgoing stream and will involve changing the sequence number and occasionally the marker bit .

Upon reception of a packet, the Select-And-Forward bridge will update that conferee's state and power signal envelope. It will then perform speaker selection by re-ranking the speakers given the updated conferee state and the last available state for the other conferees (see Section 2.6.1). If the conferee in question has been selected as a speaker, the packet will be forwarded to the other $N-1$ conferees. Otherwise, the packet will simply be dropped. While some form of stream synchronization and output packet rate control are suggested in [5], these are in general unnecessary, as each logical stream is considered and buffered separately by destination endpoints.

Because the bridge does not actually decode incoming packets, endpoints can add side information to upstream packets that specify speech energy and a VAD decision for the given packet. This allows the bridge to perform speaker selection without having to decode the packet. Alternatively, the bridge can do a partial or full decode of incoming packets in order to extract these parameters.

Packet Header Translation

A Select-And-Forward bridge performs minimal RTP packet header translation on selected (and thus forwarded) packets. The bridge acts as an RTP *translator*: the SSRC field of forwarded packets is not modified and the packet will still appear to come from its original source, rather than the bridge.

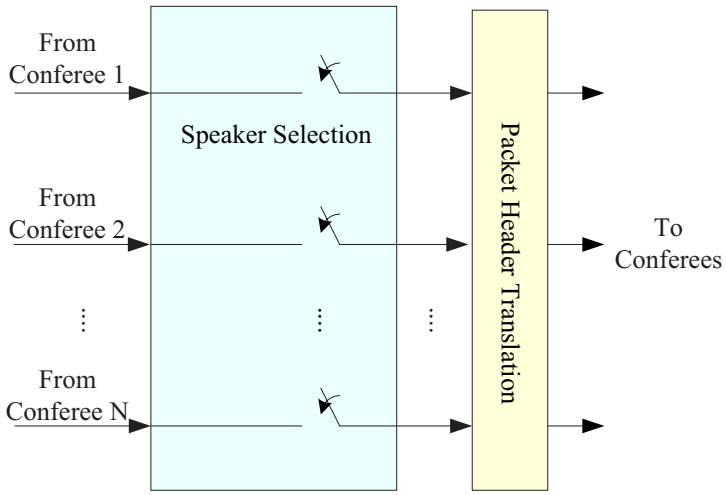


Fig. 2.12 Select-And-Forward Bridge. Outgoing packets will be sent to all conferees except for the original source of the packet.

The bridge will drop packets from a given source during periods where that source is not selected as a speaker. The bridge must therefore reassign sequence numbers on selected packets such that no sequence number gaps appear in outgoing streams. The sequence number for a given stream is only incremented when a packet for that stream is selected as speaker. The marker bit should be set on the first packet to be selected and forwarded when a non-selected stream becomes re-selected in order to mimic silence suppression [5]. Timestamps are not modified by the Select-And-Forward bridge.

Endpoint Requirements

In a Select-And-Forward conferencing scenario, endpoints will see one logical destination (i.e., the bridge) while seeing $N-1$ incoming streams and $N-1$ logical sources (one for each other conferee). Because the bridge acts only as a packet reflector, the rest of the conference is not abstracted from endpoints. Each of the $N-1$ streams needs to be dejittered and buffered separately. In addition, each stream will have a different logical source, and thus have different network delay characteristics. It should be noted that while the endpoint sees $N-1$ logical streams, on average only M will be active at any given time. Since packets will only arrive for a given stream when the conferee associated with that stream is selected as a speaker, endpoints must support silence suppression.

In general it is simplest if all endpoints in a conference controlled by a Select-And-

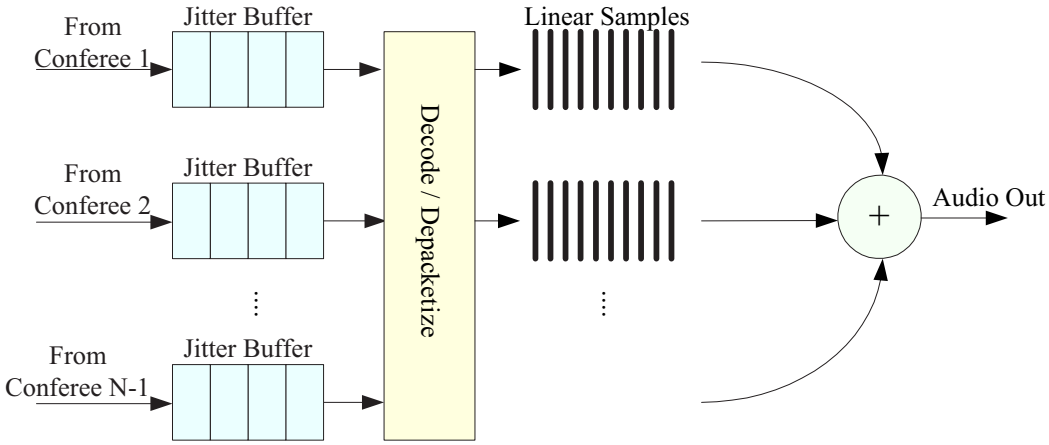


Fig. 2.13 Endpoint for use with a Select-And-Forward Bridge. All packets will arrive via the bridge. Each jitter buffer will use a separate set of network statistics in order to perform playout scheduling. In general, only M streams will be active at a given time, so most buffers will be empty.

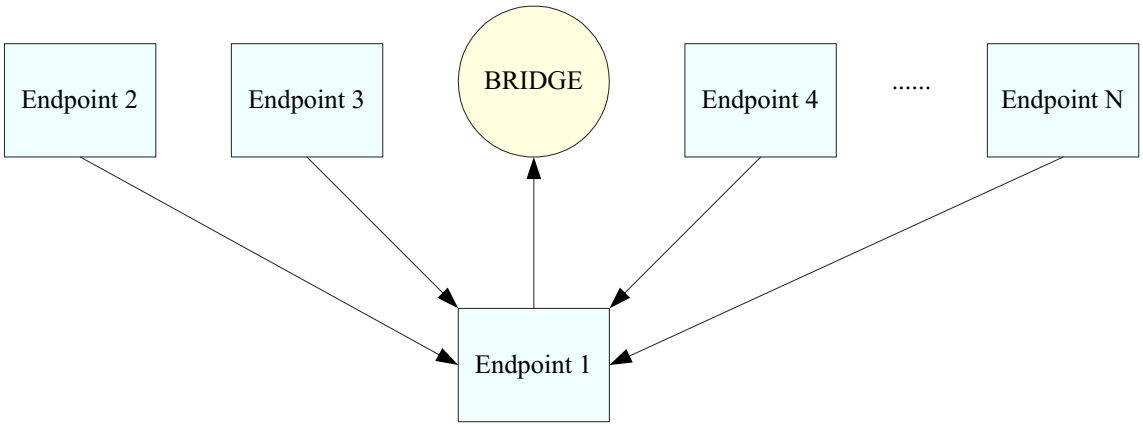


Fig. 2.14 Endpoint perspective in Select-And-Forward Conference. Each endpoint will see $N-1$ incoming streams from $N-1$ different logical sources.

Forward bridge are all using the same codec. The use of different codecs is possible so long as all codecs in use are all supported by all endpoints. Different endpoints can easily use different RTP packetization intervals, as mixing is only done at endpoints on a sample-by-sample basis. Because the Select-And-Forward bridge does not perform any stream synchronization or any mixing, it can easily support different size packets from different endpoints. The speaker selection algorithm specified in Section 2.6.1 has inherent support for comparing streams with different packet sizes.

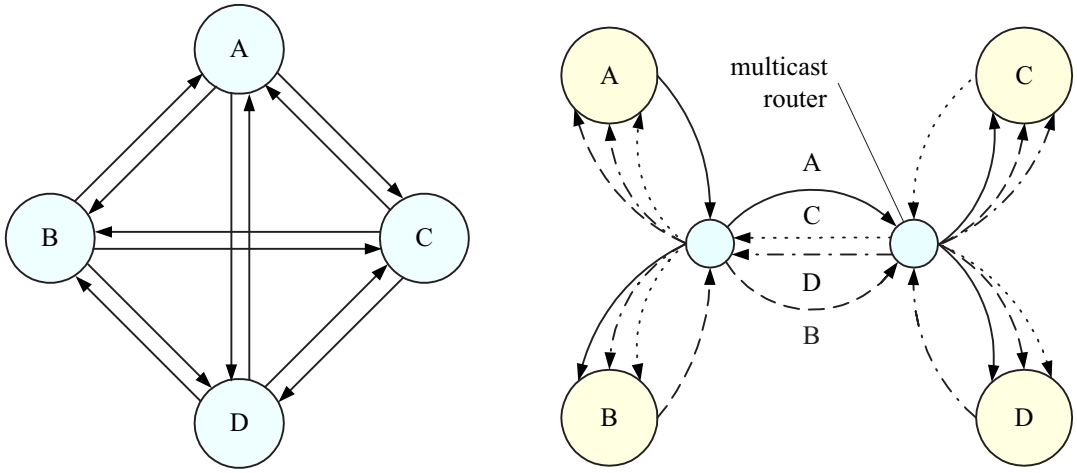
The use of state based codecs may result in state synchronization problems between source and receiver when a given talker is unselected. Consider an RTP stream between conferees A and B. When A is not selected as a speaker, frames of speech are being generated at A, but not being received at B, as they are being dropped at the bridge. When A is once again selected as a speaker, there will have been a gap in the stream of voice frames received at B from A. This will result in a state synchronization error. In most cases, periods where a given conferee is not selected as a speaker by the bridge will correspond to silence periods from that conferee, which should mitigate any perceptual effects of this state synchronization issue. Experiments done with G.723.1 have shown that state synchronization errors introduced during silence periods inject little appreciable distortion.

2.8 Other Topologies

While traditional conferencing in the PSTN is of a centralized nature, the flexibility provided by IP allows for a class of decentralized conferencing models. The absence of a bridge allows for both a reduction in the end-to-end delay experienced by packets and the elimination of any tandem encoding. Decentralized architectures require that the endpoint be able to receive and mix $N - 1$ streams.

2.8.1 Full Mesh

In a full mesh conference, each endpoint establishes a one-to-one connection with each of the other $N - 1$ endpoints. Copies of each speech packet are distributed via multi-unicasting [80]. The full mesh model requires that each endpoint pair share a common codec; however, any pair of endpoints can communicate using the codec of their choice without affecting any other conferees.



(a) Full Mesh (b) Multicast

Fig. 2.15 Decentralized conferencing models. In (a), endpoints send their audio $N-1$ times (in this case, 3). In (b), solid lines represent mixed voice streams, while other dashed lines represent single voice streams. Endpoints only send their audio once in the multicast case [5]

In a full mesh conference, each endpoint requires enough bandwidth to accommodate $N-1$ output streams and $N-1$ input streams. This bandwidth constraint generally limits the full mesh conferencing model to small conferences. The scalability of a full mesh conference is further limited by a computational load at each endpoint that increases with N.

2.8.2 Multicast

In a multicast conference, a *single* copy of each endpoint's audio is tramitted to the conference *multicast address* and each endpoint receives $N-1$ streams in return [81]. This greatly reduces the bandwidth requirements as compared to the full mesh model as the multicast network will replicate and forward each stream in a bandwidth efficient manner.

The multicast environment requires that all endpoints use a common codec. While the multicast conference scales well in terms of bandwidth requirements, the computational load on endpoints still increases with N as endpoints must still support $N-1$ incoming streams.

Chapter 3

Synchronized VoIP Conference Bridge

This chapter presents a design for a Synchronized VoIP conference bridge, similar to that published by Simard et. al. in [3], with the addition of details of novel intrastream and interstream synchronization algorithms. The Synchronized bridge provides several advantages over the Select-and-Forward bridge:

- A continuous and periodic output of M selected packets from the bridge.
- The abstraction of conference details from endpoints.
- A hierarchical design that scales well to large and/or multiple bridge conferences.
- The ability to mix or bundle selected packets, allowing for a one-to-one connection model with endpoints.
- Greater immunity to sampling clock skew.

The Synchronized bridge attempts to fully abstract conference detail from endpoints by buffering and synchronizing incoming streams. This *abstraction* allows for the elimination of any signalling between an endpoint and any other endpoint. Endpoints will only need to establish M RTP session(s) with the bridge and need not be aware of the number or locations of other endpoints participating in the conference.

The first step in this abstraction process is the abstraction of the network delay experience by a packet on the source-to-bridge path. This is accomplished by smoothing out the network delay jitter on the source-to-bridge delay path, achieved through an *intrastream* synchronization, or buffering, stage at the bridge. An *interstream* synchronization stage then aligns the buffered packets in time, such that speaker selection and/or packet forwarding is performed on all incoming streams simultaneously. This allows the bridge to map N input streams to M continuous and periodic output streams, representing selected speakers. This stream mapping procedure, coupled with translation of source identifier and timestamp fields in the RTP header, allows for the abstraction of the original source of the packet.

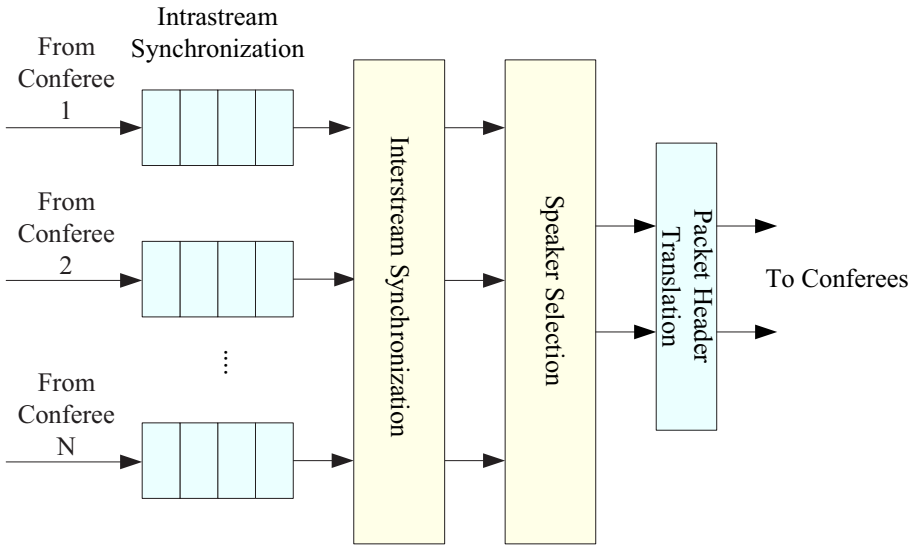


Fig. 3.1 Tandem-Free Synchronized Conference Bridge. Here $M = 2$. In tandem-free operation, incoming packets are not decoded and the M selected packets are forwarded without being mixed.

A tandem-free Synchronized bridge, forwarding M streams to endpoints, is shown in Fig. 3.1. The intrastream and interstream synchronization stages allow for easy extensibility to the bundling or mixing of selected packets.

Section 3.1 details the intrastream synchronization process, while Section 3.2 discusses interstream synchronization. Other details of the bridge, such as speaker selection and packet header translation, are discussed in Section 3.3.

3.1 Intrastream Synchronization

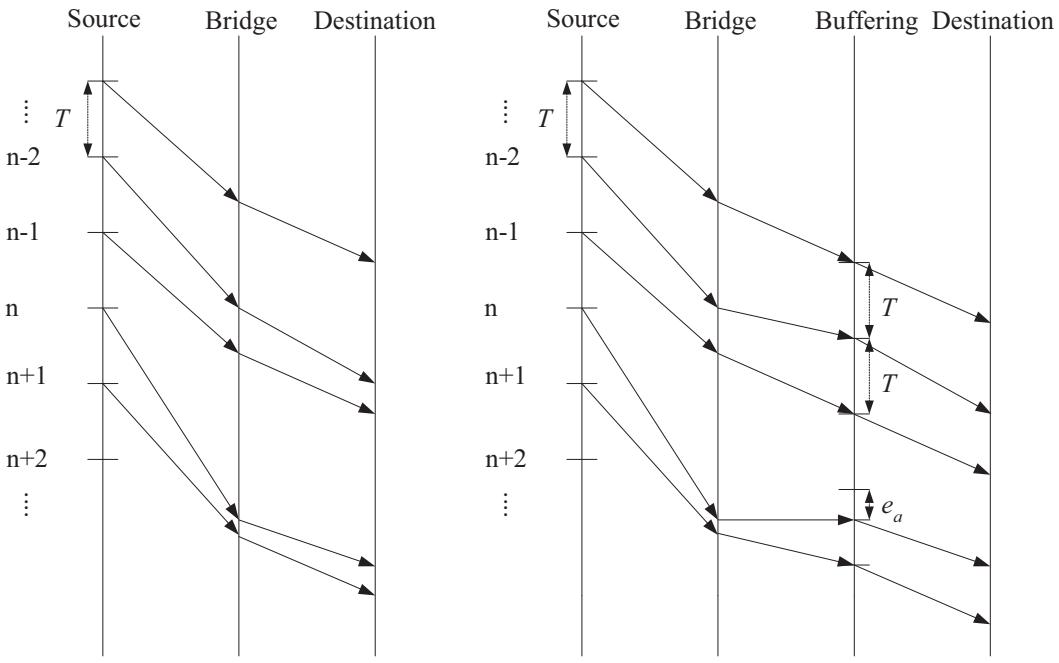
The goal of the intrastream synchronization stage is to dejitter the incoming packets so as to remove most of the network delay variation accumulated on the source-to-bridge path. This allows for the abstraction of the source-to-bridge network delay from the rest of the conference endpoints.

The buffering is done much in the same way that playout scheduling is done for endpoints (Chapter 2.4); the main difference is that instead of buffering for playout of a packet, the bridge is buffering for the speaker selection and potential forwarding of a packet. The goal in playout scheduling at an endpoint is to preserve a continuous and periodic playout stream, while the goal in bridge buffering is to preserve a continuous and periodic forwarding of selected packets from the bridge, for a given stream. The bridge buffering scheduler will assign a *forwarding time* for each packet, analogous to the playout time assigned to packets at endpoints. Fig. 3.2 shows a periodic output of packets from the Synchronized bridge, as compared to the Select-and-Forward bridge which forwards packets immediately.

While any of the playout scheduling approaches discussed in Chapter 2.4 are valid choices as a basis for a bridge buffering algorithm, a histogram based approach based on methods described in [52] was deemed best for use in bridge buffering due to its ability to be explicitly parameterized by a target late rate. Histogram-based methods do have the disadvantage of memory requirements for storing past delay values (for each incoming stream). Auto-regressive-based algorithms may be a better approach if bridge memory requirements are to be kept at a minimum.

Since the bridge is dealing with outputs in units of (forwarded) packets, its buffering constraints are different from those at the endpoints, whose output (playout) is in units of samples. The bridge essentially has a time resolution of *one packet* in adjusting a stream's forwarding time. While an endpoint may compress or expand a silence period by an arbitrary number of samples in order to adjust its playout delay, adjusting forwarding delays at the bridge by amounts less than a packet length is much more complicated[1]. The bridge buffering algorithm thus essentially becomes a fixed playout scheduler whose only adaptive capabilities are to shift the forwarding time by units of packet lengths. A

[1] Adjustments would have to be a multiple of the codec frame length and would require the forwarding of a *shortened* packet with appropriate timestamp advancement so as to inform endpoints of the shift. Any adjustments by less than a packet length would have to be common across all streams in order to maintain interstream synchronization.



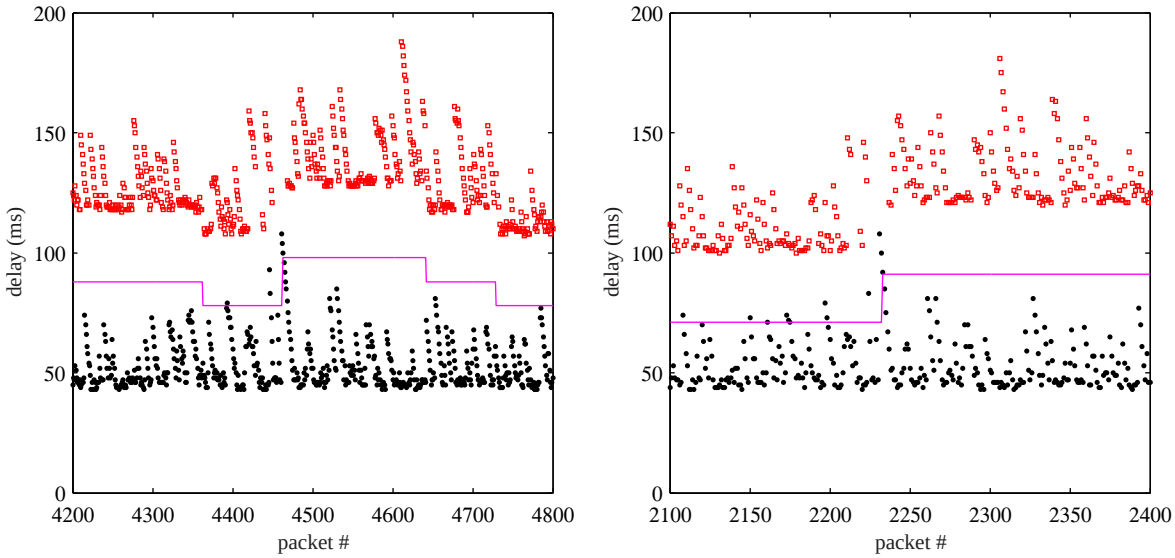
(a) Select-and-Forward Timing. (b) Timing with Bridge Buffering.

Fig. 3.2 Intrastream Synchronization. In (b), the output of selected packets from the bridge is periodic, except where packet n arrives late for its forwarding time. The late arriving packet n results in an abstraction error, e_a. While additional delay is injected into the source-to-destination path, the receive delay variability is reduced at the endpoint.

smaller RTP packetization interval gives the bridge buffering algorithm greater adaptive capabilities by giving it a finer time resolution. Any shift in forwarding time can be done on a stream only when that stream is not currently selected as a speaker, or when packets in a selected stream are carrying silence or background noise, so as not to introduce any audible distortion. Bridge intrastream synchronization with different packetization intervals is shown in Fig. 3.3.

In traditional playout scheduling, a packet arriving later than its scheduled playout time is deemed *lost*. In bridge buffering, a packet arriving later than its scheduled forwarding time can be forwarded to endpoints in the hope that it will still arrive in time for playout at the destination endpoint.

Packets arriving later than their forwarding time introduce a *delay abstraction error*,



(a) Bridge Buffering with 10ms packets.

(b) Bridge Buffering with 20ms packets.

Fig. 3.3 Bridge Buffering. The lower dots represent the delay from source to bridge. The line represents the forwarding time of each packet. The upper squares represent the arrival time of selected packets at the destination endpoint. These plots represent the same time period and network delay patterns. A target late rate of 2% and a histogram length of 150 packets are used in both cases. The 10ms packets allow for more flexibility in adjusting the forwarding time.

defined as the amount of time by which the packet arrives at the bridge after its scheduled forwarding time. Delay abstraction errors cause a deviation from the periodic output of packets from the bridge for a given stream and can cause playout scheduling errors at destination endpoints. The effect of delay abstraction error is explained in greater detail when discussing timestamp translation (Section 3.3.4).

3.1.1 Intrastream Synchronization Algorithm

The bridge buffering algorithm is parameterized by a target *late rate* analagous to a target loss rate for playout scheduling. The choice of late rate is a trade-off between buffering delay and delay abstraction error. A target late rate of zero will provide total abstraction of the

source to bridge network path for the destination endpoints, but at the cost of a potentially unbounded buffering delay. Late rates in the range of 5% to 50% were used in simulations. The choice of late rate depends on the synchronization and delay requirements, as well as the differences in network delay characteristics of the various endpoint-to-bridge paths.

Histogram-based buffering algorithms allow for tighter control (as opposed to AR or adaptive filter-based algorithms) on the actual late rate experienced by the bridge, especially as the histogram size, N, is made large. A large N results in a more accurate estimate of a given percentile delay, giving an actual late rate closer to the one targeted. A large N comes at the expense of larger bridge memory requirements (to store the previous N delay values), a longer warm-up period (to collect the first N delay values), and less flexibility in the face of rapidly changing network conditions.

The first N packets received at the bridge from a given source are discarded, and the calculated delay values from the RTP timestamps are ordered in a histogram. This constitutes the *warm-up* period. Upon reception of packet $N+1$ or any successive packet from that given source, the *target* forwarding time t_i is selected as:

$$t_i = a_i + d_l - n_i, \tag{3.1}$$

where a_i is the arrival time of packet i, d_l is the l^{th}-percentile delay taken from the histogram (where l is the target late rate), and n_i is the network delay of the current packet as calculated from RTP timestamps.

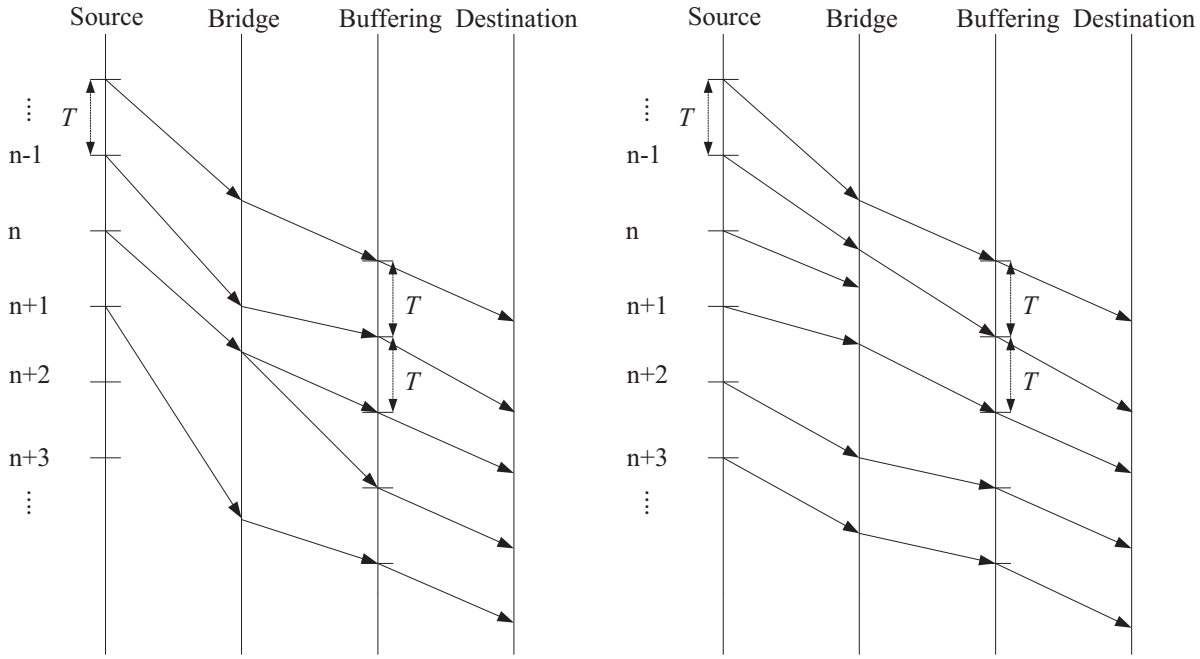
Packet $N+1$ will have a forwarding time f_{N+1}, equal to its *target* forwarding time, while others will have forwarding time dictated by the previous packet's forwarding time, so as to ensure periodic forwarding of packets:

$$\begin{aligned} f_{N+1} &= t_{N+1} \\ f_i &= f_{i-1} + T : i > N+1, \end{aligned} \tag{3.2}$$

where T is the packet length. During intervals where the stream is silent, the forwarding time of a packet in that stream can be adjusted by a multiple of the packet length[2]. Ad-

[2] Any adjustment during periods where a stream is actively talking will lead to a discontinuity at destination endpoints when these packets are forwarded — dropping or repeating a packet during silence period is not noticeable to the listener.

justments on forwarding times are done when the difference between the target (Eq. (3.1)) and fixed (Eq. (3.2)) forwarding times is greater than one half of one packet length. In these cases, forwarding times are reduced by dropping the current packet or increased by repeating the previous packet (shown in Fig. 3.4). Processing requirements can be reduced by only calculating the target forwarding time once in a while (say every 10 packets), since it is only useful in deciding whether or not to drop or repeat a packet.



(a) Repeating a packet.

(b) Dropping a packet.

Fig. 3.4 Forwarding Time Adjustment. In (a), the forwarding time is increased by T by repeating packet n, while in (b), the forwarding time is decreased by T by dropping packet n. Dropping and repeating of packets should only be done during silence periods.

Spike Detection

Like most playout scheduling algorithms, intrastream synchronization algorithms used at the bridge use a form of *spike detection*. When doing playout scheduling at endpoints, spike detection is used in an attempt to quickly adapt to the sudden increase in network delay.

The approach used in dealing with spikes at the bridge is to ignore packets that are part of a spike by not adding the network delays of those packets to the network delay histogram. The l^{th}-percentile delay estimate used in Eq. (3.1) is thus only a function of *non-spike* packet delays. Forwarding times are assigned as in the case of non-spike packets (Eq. (3.2)). Most, if not all, of the spike packets will arrive late at the bridge for their scheduled forwarding time. Any network delay spike on the source-to-bridge path is in effect *passed on* to the destination endpoints in the form of delay abstraction error. The reasoning for this approach is twofold:

- Endpoints may be better able to deal with rapidly changing network delays than bridges.
- Attempting to track network delay spikes at the bridge may result in excessive buffering delays at the bridge.

Spike mode is triggered when a packet arrives with a calculated delay that exceeds the previous packet's calculated delay by a given threshold, S_h. The spike is considered over when a packet arrives with a calculated delay that exceeds the previous *non-spike* delay by *less* than another threshold, S_l [41].

In general it is not easy to exactly achieve an actual late rate corresponding to the specified target late rate. This is because once the first forwarding time has been calculated, all subsequent forwarding times will only differ by a multiple of the packet length.The frequency of network delay spikes also affects the discrepancy between target and actual late rate as spike packets are not included in the delay histogram, but will still count as *late* packets. A longer histogram length helps achieve a more accurate characterization of network delay characteristics which leads to a smaller difference between target and actual late rates.

3.1.2 Effect of Intrastream Synchronization

The main drawback of the intrastream synchronization stage is that it adds delay. The overall delay penalty can be evaluated by measuring the additional end-to-end delay introduced as a result of buffering at the bridge. *End-to-end delay* is defined as the time between generation of a packet and playout of that packet. *Receive delay* is defined as the time between generation of a packet and reception of the packet at its final destination

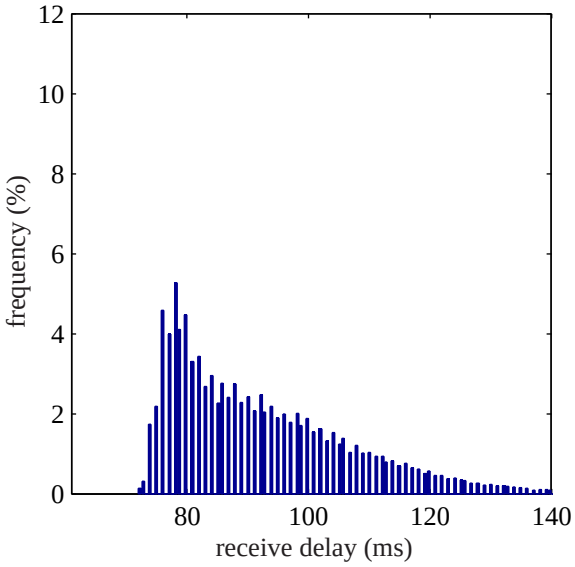
(but does not include any buffering incurred due to playout scheduling at the destination endpoint). While intrastream synchronization will increase the average *receive delay* experienced by packets, the *end-to-end delay* penalty will in general be less than the amount of time for which a packet is buffered at the bridge. Since the delay jitter from the source to bridge path and delay jitter from the bridge-to-endpoint path are, in general, additive, bridge buffering will decrease the overall delay jitter as compared to the Select-and-Forward bridge, thus reducing buffering requirements at destination endpoints.

Fig. 3.5 shows the effects of intrastream and interstream synchronization on receive delay. Fig. 3.5(a) shows the receive delay distribution for a given endpoint-bridge-endpoint network path when using a Select-and-Forward bridge, while Fig. 3.5(b) shows the receive delay distribution for the same path under the same network conditions after having performed intrastream synchronization at the bridge. One can see that the lowest delay packets in Fig. 3.5(a) have their receive delay increased by the intrastream synchronization process (notice the large spike in the histogram shown in Fig. 3.5(b) around 90ms), while the higher delay packets are little affected by the intrastream synchronization stage, as can be seen by the similarities in the distribution of high receive delays.

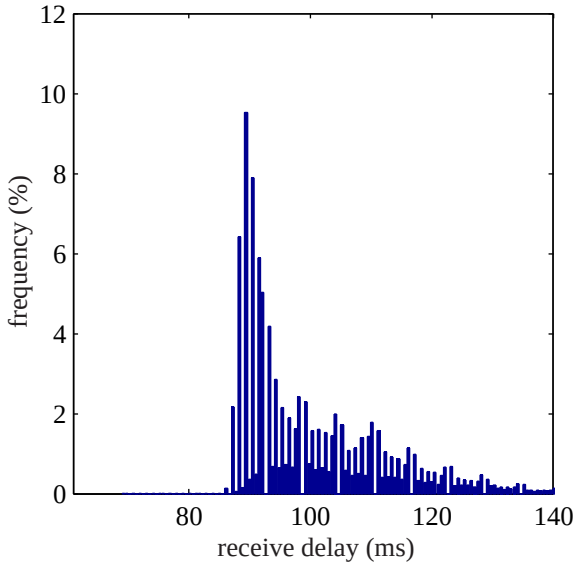
Since the playout delay of a packet at the endpoint is mostly a function of the highest-delay packets, the effect on the end-to-end delay is mitigated somewhat. While the average receive delay will be substantially higher, the receive delay of the longest delayed packets will not be affected much, if at all. Endpoints using histogram based playout algorithms, especially those with large N, will see a low overall delay penalty because they only consider the longest 5% or less of receive delays when scheduling playout, which are little affected by bridge buffering. Destination endpoint playout algorithms that are packet adaptive will incur a larger delay penalty because of their ability to react quickly to changing delay; in these cases, packets with lower receive delays are also taken into account when doing playout scheduling.

3.2 Interstream Synchronization

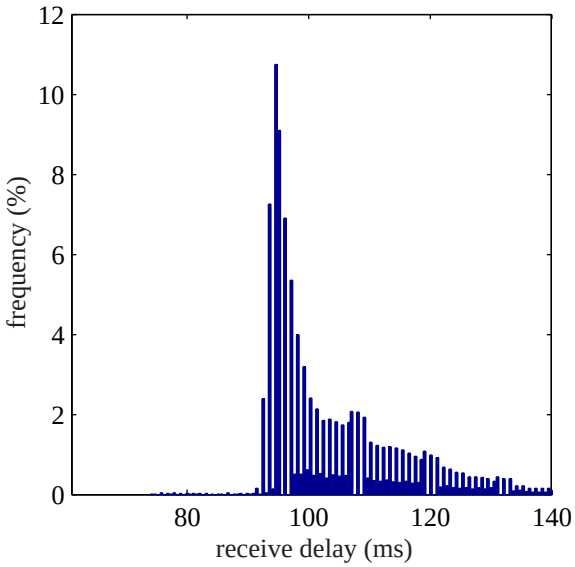
The goal of interstream synchronization is to have common forwarding times across all incoming streams. While the intrastream synchronization stage creates a periodic forwarding of packets from within a given stream, the interstream synchronization stage ensures that the forwarding of packets will be periodic across all streams. After interstream synchro-



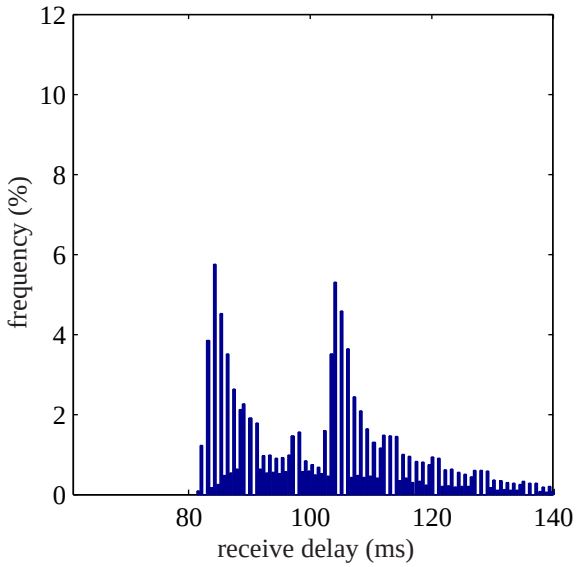
(a) Receive delay distribution using Select-and-Forward bridge.



(b) Receive delay distribution after *Intra*stream Synchronization stage.



(c) Receive delay distribution after *Intra*stream and *Inter*stream Synchronization stage.



(d) Receive delay distribution after *Intra*stream and *Inter*stream Synchronization stage, using a different master stream for synchronization.

Fig. 3.5 Effect of Intrastream and Interstream Synchronization on Receive Delay. (a) shows the receive delay distribution when using a Select-and-Forward bridge. (b) shows the receive delay distribution for the same path and network conditions after performing *intra*stream synchronization only. (c) and (d) show two possible receive delay distributions after performing *intra*stream *and inter*stream synchronization.

nization, packets in all incoming streams will have common forwarding times.

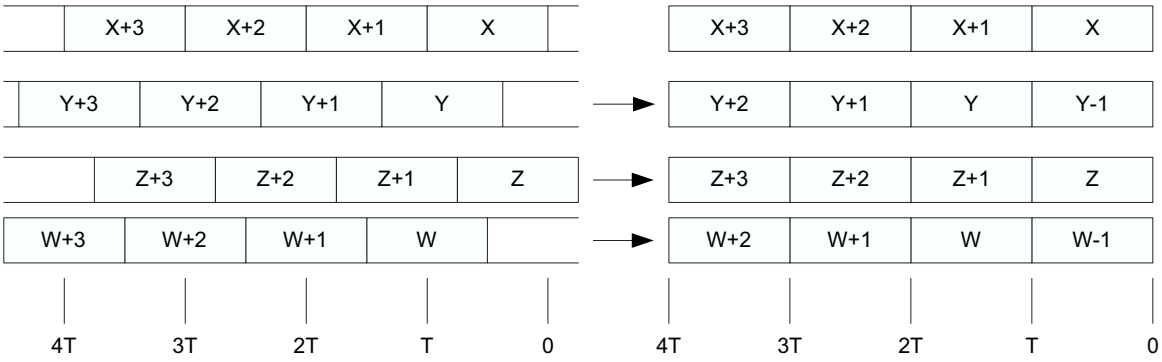


Fig. 3.6 Interstream Synchronization. Buffered packets on the left are synchronized to the master stream (in this case the top stream). The synchronization stage will introduce an additional buffering delay in the range of 0–T, where T is the packet length.

3.2.1 Interstream Synchronization Algorithm

Interstream synchronization is achieved in much the same way as suggested in [67]. The stream that is currently selected as primary speaker is chosen as the master stream. Other streams are buffered such that the packets in that stream have forwarding times that are a multiple of a packet length different from the forwarding times of the master stream. If a packet in a slave stream has forwarding time f_i as calculated after the buffering stage, then its synchronized forwarding time s_i is going to be calculated as:

$$s_i = f_i + ((m_j - f_i) \mod T), \tag{3.3}$$

where m_j is the forwarding time of a packet in the master stream[3], and T is the packet length.

Once a stream has been synchronized to the master stream, the synchronized forwarding time, s_i, can be used as a basis for any further intrastream synchronization:

$$f_{i+1} = s_i + T. \tag{3.4}$$

In general, once the interstream synchronization is first performed, f_i and s_i will be the

[3]The forwarding time of any packet in the master stream that is currently being buffered can be used.

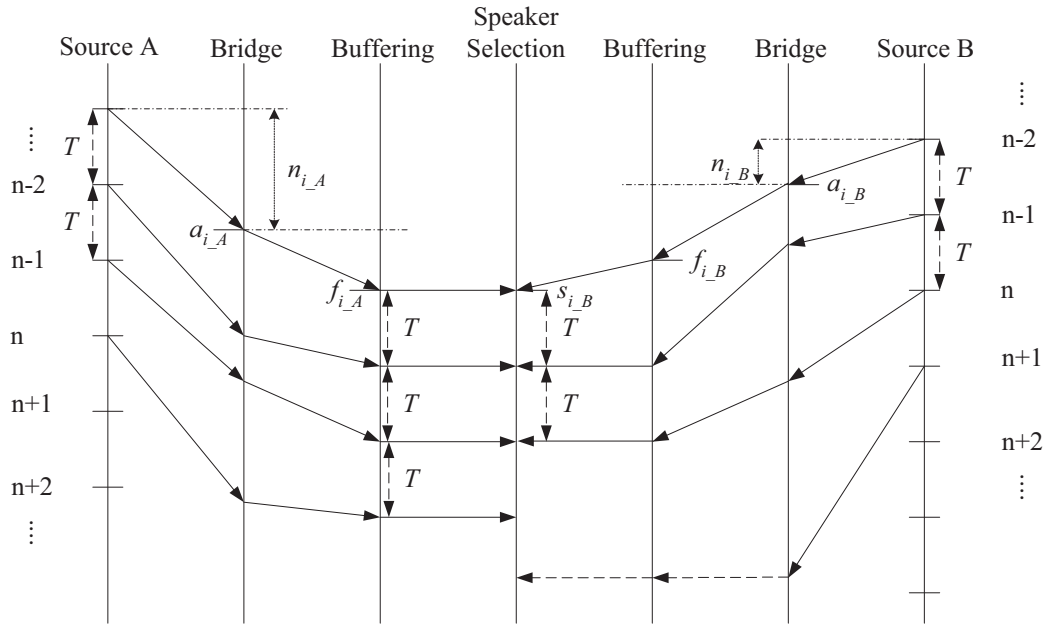


Fig. 3.7 Intrastream and Interstream Synchronization. Here the stream from source A acts as the master stream, while the stream from source B is synchronized to it. Subsequent forwarding times for stream B are based on the synchronized forwarding time, so no further interstream synchronization is required. The last packet from source B is late for its scheduled forwarding time.

same for subsequent packets. Timing for the intrastream and interstream synchronization processes are shown in Fig. 3.7.

When a new primary speaker is selected, the stream for that new speaker is selected as the master stream, and the old master stream becomes a slave stream. Because the new and old master streams will have already been synchronized, and any subsequent buffering is done based on the synchronized forwarding times, this switch will be transparent.

3.2.2 Effect of Interstream Synchronization

The interstream synchronization will have no effect on the master stream. The slave streams will however be subjected to additional buffering so as to line up packet boundaries with the master stream. The additional receive delay incurred by slave streams is dependent on the nature of the network delay distributions. Consider a slave stream with initial forwarding time (before interstream synchronization), f_i. The corresponding synchronized

forwarding time (after interstream synchronization), s_i, is calculated as in Eq. (3.3), and can be expressed as

$$\begin{aligned} s_i &= f_i + ((m_j - f_i) \mod T) \\ &= f_i + d_s, \end{aligned} \tag{3.5}$$

where d_s is the buffering delay incurred because of interstream synchronization and falls in the range $0 - T$. All subsequent forwarding times will be will be restricted to

$$\begin{aligned} f_{i+1} &= s_i \pm nT \\ &= f_i + d_s \pm nT. \end{aligned} \tag{3.6}$$

The interstream synchronization effectively changes the initial estimate of the l^{th}-percentile delay for that stream from f_i to $f_i + d_s$, giving a poor estimate of the l^{th}-percentile delay and potentially injecting additional delay, while providing a lower late rate at the bridge. One can observe this effect in Fig. 3.5(c), where the spike in the histogram is at a slightly higher delay than in Fig. 3.5(b). The difference between the location of these histogram spikes is equal to d_s (see Eq. (3.5)).

In other cases, the interstream stage may lower the average bridge buffering delay. In Fig. 3.5(d), the buffering delay incurred due to interstream synchronization, d_s, is larger than that in Fig. 3.5(c). Two spikes in the histogram can be observed in the delay histogram, with the spikes lying one packet length (20ms) apart. This suggests that the *target* forwarding delay (Eq. (3.1)) fluctuates between being closer to one histogram spike and then the other. Since the bridge can only adapt in units of packet lengths, it can only go back and forth between the two buffering points, even if the *optimum* buffering point usually lies near the midpoint of the delays represented by the histogram spikes.

3.3 Tandem-Free Synchronized Bridge

3.3.1 Speaker Selection

After the interstream synchronization stage, all input streams to the bridge will have common and periodic forwarding times. These common forwarding times serve as triggers for performing speaker selection and updating speaker rank among conferees. At a given forwarding time, f_i, all streams will have their power signal envelope, $\hat{E}_i$, updated with the signal power of the current packet as in Eq. (2.21). If a packet is *late* for its scheduled forwarding time, the last available value for the power signal envelope is used for speaker selection for that stream, and the power signal envelope is updated once that late packet does finally arrive.

Once the power signal envelopes have been updated, the speakers are re-ranked, according to the barge-in constraints detailed in Section 2.6.1. Packets from the M selected speakers are then forwarded to endpoints, while packets from the remaining streams are dropped. If a packet is late for a stream that has been selected (and is therefore due to be forwarded), the packet will be forwarded immediately upon arrival. The handling of late packets will be discussed in greater detail in Section 3.3.3.

3.3.2 Packet Header Translation

A synchronized bridge acts as an RTP *mixer*. Packets selected as speakers will have their timestamp, sequence number and source identifier (SSRC) fields modified before being forwarded to conference endpoints. The translation of the source identifier allows for the abstraction of the original source of the packet, while timestamp translation provides abstraction of the network delay incurred on the source to bridge path.

Source Translation

The synchronized speaker selection procedure provides a mapping of N input streams to M output streams. Each of the M output streams has a source identifier (SSRC in the RTP header) assigned to it. The easiest way to map selected packets to an output stream is to have each of the M output streams represent a *speaker rank*. When an input stream is selected as a speaker, the SSRC field in that packet will be modified so as to represent the speaker rank assigned to that stream. In this way streams outgoing from

the bridge will represent the primary speaker, the secondary speaker, and so on. This is shown in Fig. 3.8(b). This approach is useful for instances in which an endpoint does not have the ability to receive and/or mix multiple streams, as that endpoint can still receive an acceptable representation of the conference by only considering the primary speaker stream.

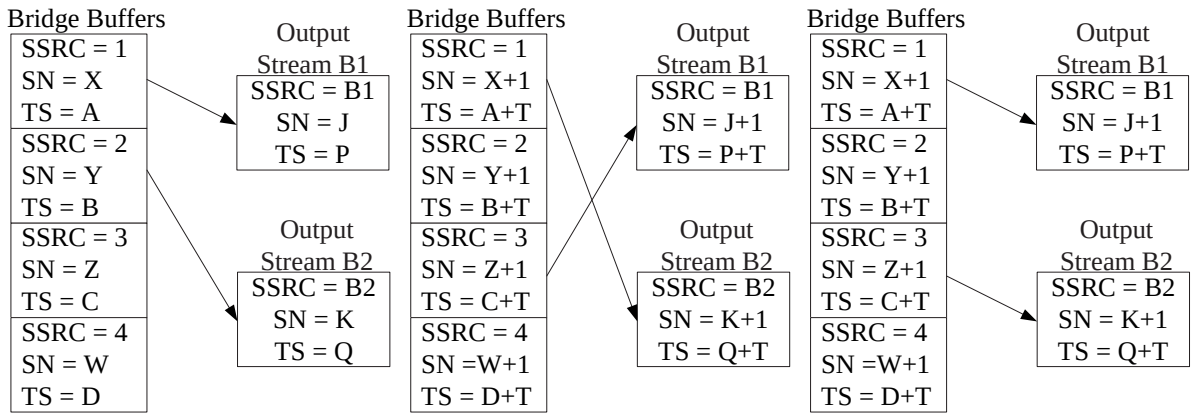
Alternatively, the bridge can map selected input streams to output streams in a manner that minimizes transitions of original sources on a given output stream. Source identifiers on outgoing packets would in this case not represent a particular speaker rank but rather be assigned as output *slots* become free. In this way, an input stream that was selected as the primary speaker and then becomes the secondary speaker, would retain the same source identifier on outgoing packets. A change in the mapping of an input stream to an output stream would thus only occur if a stream went from selected (regardless of speaker rank) to unselected. This approach, shown in Fig. 3.8(c), reduces switching in the mapping of an input stream to an output stream, which can benefit playout scheduling at endpoints (see Section 3.3.4) as well as minimize codec state synchronization errors at endpoints (see Section 3.3.6).

Timestamp Translation

The RTP timestamp on outgoing packets is changed to reflect the time at which the packet was *scheduled to be forwarded* from the bridge. This timestamp value will come from a counter tied to a bridge clock running at the speech sampling rate (8kHz in this case). If a selected packet has arrived at the bridge before its scheduled forwarding time, the timestamp will correspond to the actual time at which it left the bridge. In cases where a selected packet is late for its scheduled forwarding time, the timestamp will correspond to the time at which it would have been forwarded had it not arrived late. This ensures compliance with the RTP standard by forcing any two consecutive outgoing packets to have timestamps that differ by the number of sampling instants in a packet [8].

Sequence Number Translation

The M streams leaving the bridge will be continuous, so there will be no issues with sequence number gaps as experienced in the Select-and-Forward case. The sequence number for each of the M streams is simply incremented once every packet period, such that each



(a) Packet Header before Speaker Transition.

(b) Packet Header Translation using Speaker Rank.

(c) Packet Header Translation minimizing stream switching.

Fig. 3.8 Packet Header Translation on speaker transition. In (a), Stream 1 is the primary speaker and Stream 2 is the secondary speaker. On the next packet, Stream 3 becomes the primary speaker and Stream 1 the secondary speaker. Source translation based on speaker rank is shown in (b), while in (c) stream switching is minimized by keeping input Stream 1 mapped to Output stream B1, even though its speaker rank has changed.

outgoing packet in each stream will have a sequence number one greater than the previous packet forwarded for that stream. The initial sequence number for each stream should be chosen randomly, as specified in the RTP standard.

Contributing Sources

The CSRC field in the RTP header should be updated to indicate the original source(s) of the packet[4]. This information is only used for gathering statistics or for updating the codec state. It is not used to identify the stream for purposes of playout scheduling (this is done using the SSRC field).

[4]Multiple original sources will be specified if mixing is performed at the bridge.

3.3.3 Late Packet Management

The mapping of N incoming streams to M outgoing streams requires an adherence to the requirements on an RTP mixer [8], namely that exactly M packets (one for each speaker rank) must be forwarded every packet period[5]. The bridge must ensure that there is no occurrence of two input streams being mapped to the same output stream for a given forwarding time, which would result in two packets bearing the same SSRC and sequence number, a violation of the RTP standard. These double-mapping scenarios can occur on speaker transitions when a selected stream has late-arriving packets. The bridge must keep a record of selected packets that are outstanding to ensure that the appropriate input stream is mapped to the appropriate output stream for every forwarded packet.

3.3.4 Delay Abstraction Error

Delay abstraction error was defined as the amount by which a selected packet is late for its scheduled forwarding time at the bridge in Section 3.1. Alternatively, it can be defined as the difference between the RTP timestamp and the value of the timestamp counter at the actual time it left the bridge.

From an endpoint's perspective, the delay abstraction error will result in a difference between the delay as calculated from RTP timestamps, and the actual delay experienced by a packet on the bridge-to-destination path. For a packet forwarded at its scheduled forwarding time (i.e., a packet that does not arrive late at the bridge), the *calculated* network delay, r_i, at an endpoint will be:

$$\begin{aligned} r_i &= a_i - f_i \\ &= b_i + t_{off}, \end{aligned} \tag{3.7}$$

where a_i is the the local sampling clock counter at the time of arrival of packet i, f_i is the RTP timestamp corresponding to the scheduled forwarding time at the bridge, b_i is the actual delay experienced by the packet on the bridge to endpoint path, and t_{off} is the offset between bridge and endpoint sampling clocks. It can be seen in this case that the

[5] In the event that a selected packet is lost between source and bridge, then no packet is forwarded for that speaker for that interval.

calculated delay is dependant only on the bridge-to-destination endpoint network delay, and that the source to bridge path is fully abstracted, irrespective of the packet's original source.

Fig. 3.9 illustrates a seamless switching of speakers, where the effect of changing primary speaker (at around packet 2215) is transparent to the endpoint. In Fig. 3.9(b), network delays calculated from timestamps at the destination are exactly equal to the network delay on the bridge-to-destination path. Fig. 3.9(c) shows the equivalent calculated delay as would be seen by the Select-and-Forward case (i.e., if no synchronization is done at the bridge). The delay variability as seen by the endpoint is much higher, and there is a jump in calculated delay at the speaker transition. It should be noted that Fig. 3.9(c) is just shown to illustrate the abstraction of the source-to-bridge network delay path in Fig. 3.9(b). In reality, the Select-and-Forward bridge would not multiplex different input streams into a *speaker* stream, as shown here.

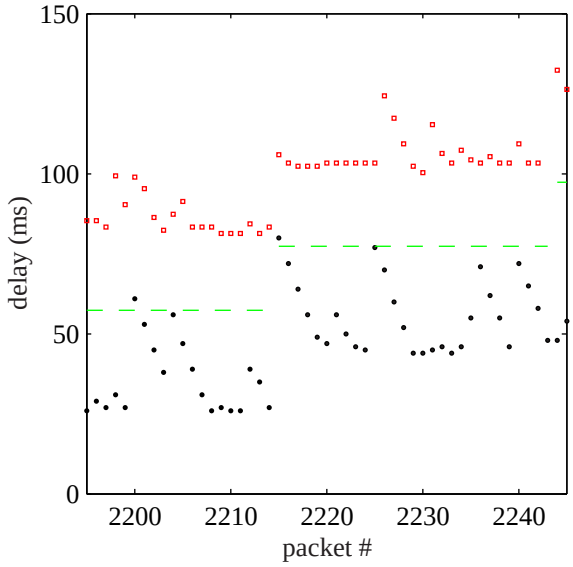
In cases where a packet arrives late at the bridge for its scheduled forwarding time, the calculated network delay, r_i, at the endpoint will be:

$$\begin{aligned} r_i &= a_i - f_i \\ &= b_i + t_{off} + a_e \\ &= b_i + t_{off} + (n_i - d_l), \end{aligned} \tag{3.8}$$

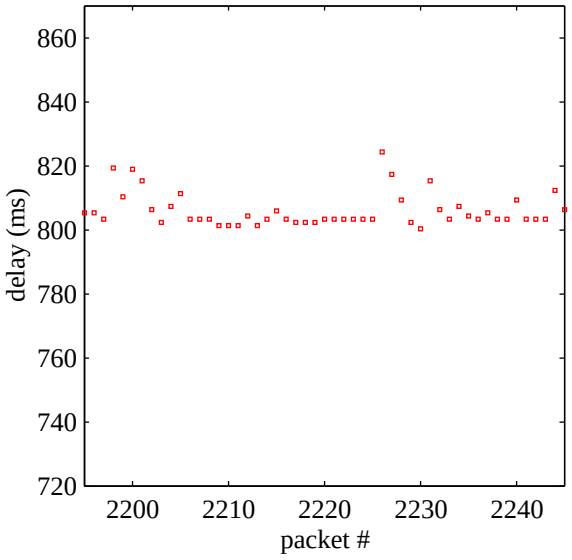
where a_e is the delay abstraction error, n_i is the network delay between source and bridge, and d_l is the l^{th}-percentile delay for the source-to-bridge path in question, as calculated from a delay histogram at the bridge. In this case the calculated delay is both a function of the delay on the bridge-to-destination path and that from the source-to-bridge path. The source-to-bridge path is thus not fully abstracted from the destination endpoint when packets arrive at the bridge later than their scheduled forwarding time.

Effect of Delay Abstraction Error

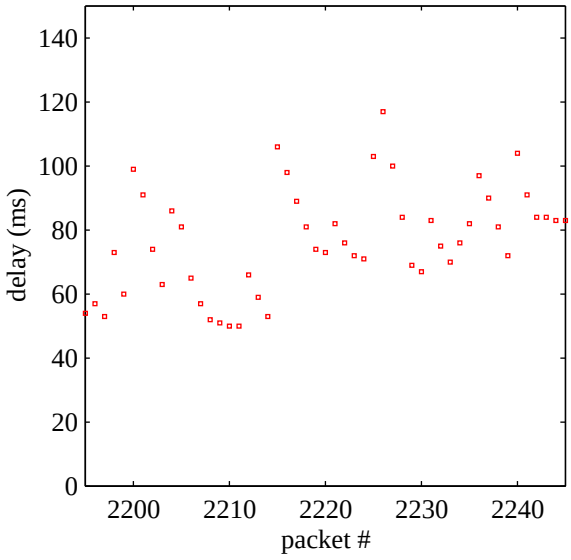
Delay abstraction error can cause playout scheduling problems at endpoints when speaker transitions occur while one of the streams involved in the speaker transition has packets arriving late at the bridge. When a network spike occurs on the source-to-bridge path, the



(a) Delay and Buffering Plot for Primary Speaker to Endpoint A.

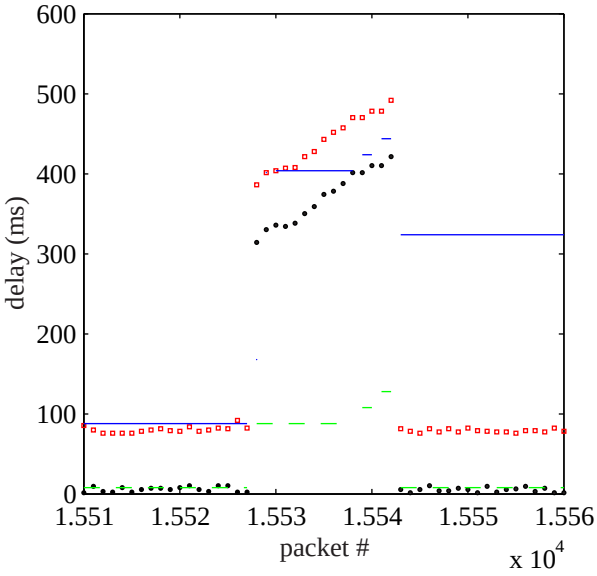


(b) Delay Calculated from Timestamps at Endpoint A.

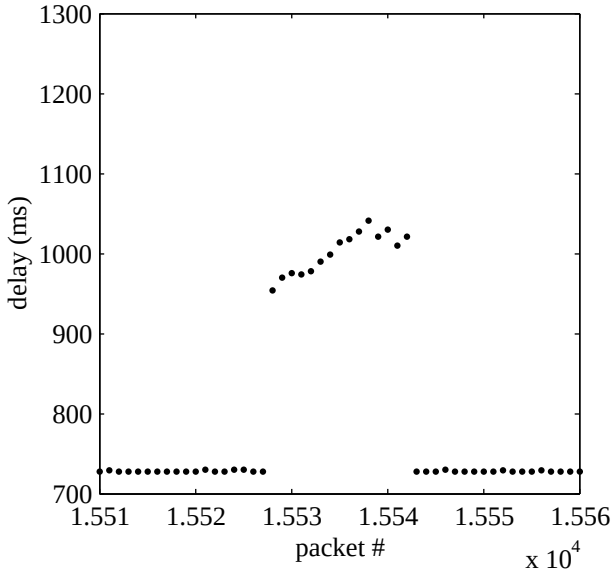


(c) Delay as would be Calculated from Timestamps at Endpoint A in the Select-and-Forward case.

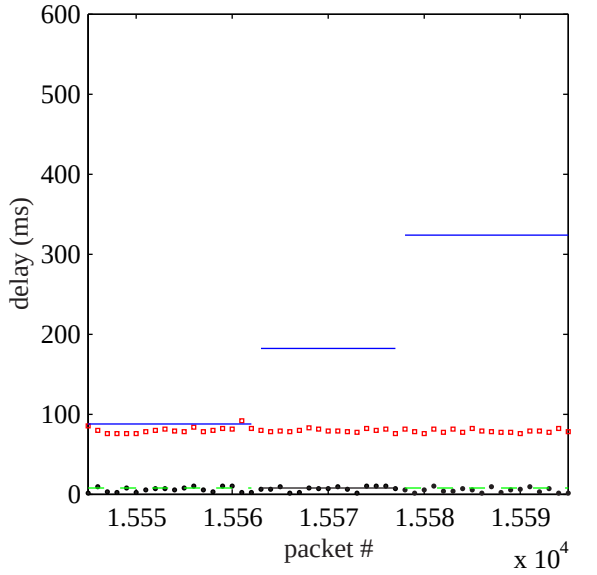
Fig. 3.9 Speaker Transition with no delay abstraction error. In (a), bottom dots represent delay from source to bridge. The bottom dotted line represents the target forwarding time (and translated timestamp) of packets from the bridge while the small squares represent the total receive delay of packets arriving at the endpoint. The original source of the primary speaker switches around packet 2215. In (b), there is a timestamp offset of 720ms (t_{off} = 720ms) between endpoint and bridge. (c) shows the delay calculated from timestamps in the Select-and-Forward case.



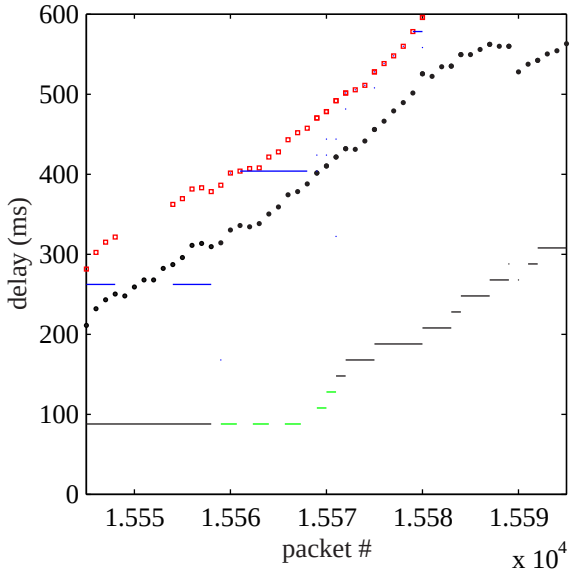
(a) Delay and Buffering plot for Primary Speaker to Endpoint A.



(b) Delay Abstraction Error plus offset(t_{off}) for Primary Speaker to Endpoint A.



(c) Delay and Buffering plot for Endpoint B to Endpoint A.



(d) Delay and Buffering plot for Endpoint C to Endpoint A.

Fig. 3.10 Effect of Delay Abstraction Error. Plot a) is for the primary speaker stream ending at endpoint A. Plots c) and d) show packets that originated at endpoints B and C, respectively, and that arrived at endpoint A. Packet numbers for c) and d) are offset by 35 packets from a) due to timestamp translation at the bridge. Dotted (green) lines represent the forwarding times of packets selected as primary speaker.

delay as calculated at the destination endpoint from timestamps will be mostly a function of the source-to-bridge network delay, n_i. This will result in the perceived network delay between bridge and destination endpoint being higher than the actual delay between the bridge and destination endpoint (from the destination endpoint's perspective) and result in unnecessary additional buffering delay when a speaker transition occurs.

The effects of delay abstraction error are illustrated in Fig. 3.10. Packets that are part of the primary speaker stream going to Endpoint A are shown in Fig. 3.10(a). These primary speaker packets originate from original sources Endpoint B (packets 15510–15527 and packets 15543–15560) and Endpoint C (packets 15527–15542). All packets originating from Endpoints B and C are shown in Fig. 3.10(c) and Fig. 3.10(d) respectively[6]. Packet numbers are offset in these two figures as compared to Fig. 3.10(a) due to packet number translation done at the bridge.

The primary speaker switches from Endpoint B to Endpoint C in Fig. 3.10 in the middle of a rapid rise in network delay on the Endpoint C-to-bridge path. From the perspective of the receiving Endpoint A (Fig. 3.10(a)), this looks like a sudden jump in delay as packets prior to 15542 in the primary speaker's stream were originating from Endpoint B. This sudden jump (as opposed to a more gradual one that would have occurred if Endpoint C had been the primary speaker all along) causes a burst of lost packets at Endpoint A.

When the primary speaker switches back to Endpoint B from Endpoint C (at packet 15543 in Fig. 3.10(a)), the stream is grossly overbuffered at Endpoint A because the prior packets originating from Endpoint C resulted in a large delay abstraction error (shown in Fig. 3.10(b)), causing calculated delays (as calculated in Eq. (3.8)) that were mostly dependent on the delays from the source-to-bridge path of Endpoint C. The *perceived* network delays of the bridge-to-destination are much higher than the *actual* delays because of this delay abstraction error. Because the playout scheduling decision is based on the prior calculated delays, the l^{th}-percentile delay is much higher than it should be.

The scenario presented in Fig. 3.10 is a fairly extreme case in which the primary speaker transitions from an endpoint with a very low delay source-to-bridge path to one with a very high and highly variable delay source-to-bridge path. In many cases, delay abstraction error does not cause significant problems during speaker transitions. If source-to-bridge

[6]These two plots represent *virtual* streams as opposed to logical streams — all packets would be translated at the bridge and become part of one of the *speaker* streams rather than maintain the source identifier of the original source. As such, only packets selected as part of the primary speaker stream will appear in both Fig. 3.10(a) and Fig. 3.10(c) or Fig. 3.10(d)

network delay variations for selected streams are similar, then speaker transitions can still be transparent to endpoints, even in the face of constant delay abstraction error.

Differences in network delay variability among the source-to-bridge paths of the various conferees dictates the amount of delay abstraction error that can be tolerated without causing too many playout scheduling errors at destination endpoints. This tolerance will in turn influence the choice of target late rate at the bridge. As the target late rate is lowered, delay abstraction error will decrease in size and frequency, at the expense of additional bridge buffering delay.

3.3.5 Bundling

The synchronization of incoming streams allows for the bundling of multiple speech packets into one RTP packet and the maintenance of a one-to-one connection model between bridge and endpoints. The bundling operation requires RTP protocol extensions but can simplify signalling requirements.

Late Packet Management

Bundling complicates management of late packets, as a decision has to made on whether to wait for selected packets that are due to be forwarded but have not yet arrived at the bridge, or to send those packets that have already arrived while discarding any late packets.

One approach is to wait for late packets if those packets belong to the *primary* speaker (the speaker with the highest rank in terms of signal power envelope). If a primary speaker's packet is late, then the other selected packet(s) are held at the bridge until the arrival of the primary speaker's packet, at which point all selected streams are bundled and forwarded [3]. If a non-primary speaker's packet is late, then the other selected packet(s) are bundled and forwarded immediately, and the late packet is dropped once it arrives at the bridge.

Another approach is to consider the signal power envelope of streams who have selected packets that are late. If streams with packets outstanding have a signal power envelope below a given threshold, then late packets from that stream need not be waited for and can be dropped.

An alternative to ignoring contributions of late arriving packets is to use some form of packet loss concealment algorithm at the bridge to approximate the contributions of a late arriving speaker [3]. This approach requires decoding of previous packets for that stream.

3.3.6 Endpoint Requirements

Endpoints participating in a conference managed by a synchronized bridge need only buffer M incoming streams, each of which will come from the same logical source (the bridge). Endpoint stream buffering is thus independent of the number of conference participants, and depends only on the number of selected speakers. Network delay statistics can (but need not) be shared across the M streams when scheduling packets for playout, as all network delays incurred on the source-to-bridge paths are abstracted by the bridge. Endpoints do need to be able to support the mixing of the M streams.

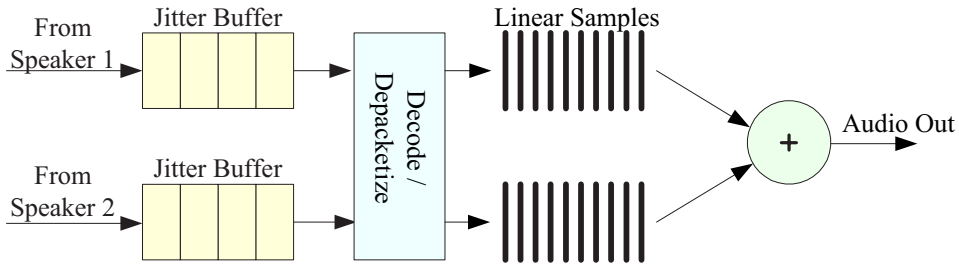


Fig. 3.11 Endpoint for use with a Tandem-Free Synchronized Bridge [79]. Here $M = 2$.

Codec Requirements

All endpoints in a conference controlled by a synchronized bridge should use the same codec. While it is possible to support multiple codecs, this would complicate decoding as packets within the same logical stream will have been encoded using different coders. Multiple codec support also requires that all endpoints be able to decode packets encoded with any coder used by any endpoint in the conference.

When using state-based codecs, state synchronization errors will be introduced on speaker transitions, as consecutive packets in a logical stream will originate from different original sources (and thus have unrelated codec states). Most speaker transitions occur during silence periods and state synchronization errors in these cases inject little audible distortion. Some transitions, particularly on the primary speaker stream, will introduce distortion due to codec state synchronization errors.

One method to reduce state synchronization errors is for endpoints to keep separate codec state for all other $N-1$ conferees. The contributing source field (CSRC) in the RTP

header can be used to determine the original source and dictate the codec state to be used in decoding that packet. While this approach does add some complexity, there is no extra signalling required to alert endpoints of other conferees' presences. Endpoints can simply add *known* conferees as they check CSRC fields of incoming packets.

The method by which the bridge translates source identifiers, i.e., the SSRC field in the RTP header, can also impact the effect and frequency of codec state synchronization errors. If the bridge assigns source identifiers based on speaker rank, then there will be more transitions for a given output stream. When the primary speaker changes, the primary speaker stream will have a transition (as the new primary speaker will come from a different source). This is especially important when dealing with the primary speaker, as it introduces the greatest probability of audible distortion because of the greater likelihood that both the previous and new primary speakers will not be in silence periods.

If source identifiers are assigned to outgoing packets so as to minimize stream switching, then the effect of state synchronization errors can be mitigated. In these cases, a transition on a speaker stream will only occur when there is a transition from speaker to non-speaker (as opposed to simply changing rank of speaker). This both reduces the number of transitions and their perceptual impact on the listener. Source translation methods were shown in Fig. 3.8.

The synchronization stage at the bridge limits the flexibility of the RTP packetization interval to be used. A synchronized conference with endpoints using different packetization intervals will need to use the lowest common multiple of all packet sizes as an equivalent logical packet size on which to do synchronization. This will limit the time resolution of speaker selection. In general it is easiest if all endpoints agree on a common RTP packetization interval.

Chapter 4

Evaluation and Results

VoIP Conference bridges can be evaluated based on performance, scalability and implementation complexity. Performance evaluations consider delay, packet loss, quality of speech, voice synchronization, and speaker selection accuracy. Scalability evaluations investigate how well a conferencing system scales to large and/or multi-bridge conferencing scenarios and how well a conferencing system can adapt to a dynamic set of conference participants. Complexity evaluations discuss loads and requirements placed on bridges, as well as those placed on endpoints participating in a conference controlled by a given bridge type.

This chapter present results of performance evaluations of the Synchronized bridge presented in Chapter 3 and of the Select-and-Forward bridge presented in Section 2.7.1. Evaluations were done using a conference simulator developed for this thesis. Results for the two bridge types were compared to determine the delay penalty attributable to performing stream synchronization at the bridge, as well as to compare their general performance. Delay and packet loss evaluations are presented in Section 4.3, while speaker selection accuracy and voice synchronization are evaluated in Sections 4.4 and 4.5. A qualitative assessment of simulated conference audio outputs is detailed in Section 4.6. The relative effects of RTP packetization interval are analyzed in Section 4.7, and robustness to sampling clock skew between endpoints is investigated in Section 4.8. Finally, scalability and complexity issues are discussed for both bridge types in sections 4.9 and 4.10.

4.1 Conference Simulator

A conference simulator was designed and coded in C, capable of evaluating prototype bridges in a controlled and configurable environment. Actual network delay traces were used to model delays between simulator endpoints and simulator bridges. Audio was taken from a four person conference and isolated for each conferee, such that the sum of the four isolated audio files represents the conference in a zero delay environment. These synchronized audio inputs for simulator endpoints provide voice stimulus to the simulator bridge for the purposes of speaker selection, and allow for the reconstruction of representative output audio as would be heard at conference endpoints.

The simulator uses a global event clock that increments the local sampling clocks of simulator endpoints and bridges. Endpoint sampling clocks trigger the packetization of linear audio samples from input audio files at intervals specified by the packet size. Audio packets are then "delayed" by an amount taken from the network trace before "arriving" at the simulator bridge. The bridge processes incoming packets (synchronization, speaker selection, header translation and forwarding/dropping) and forwarded (selected) packets are then delayed by an amount specified by different network delay traces before "arriving" back at simulator endpoints. Packets arriving back at endpoints are scheduled for playout and combined to form an audio sum[1]. In addition, network and buffering delay data is collected for each packet and dumped to a file for later processing and/or plotting.

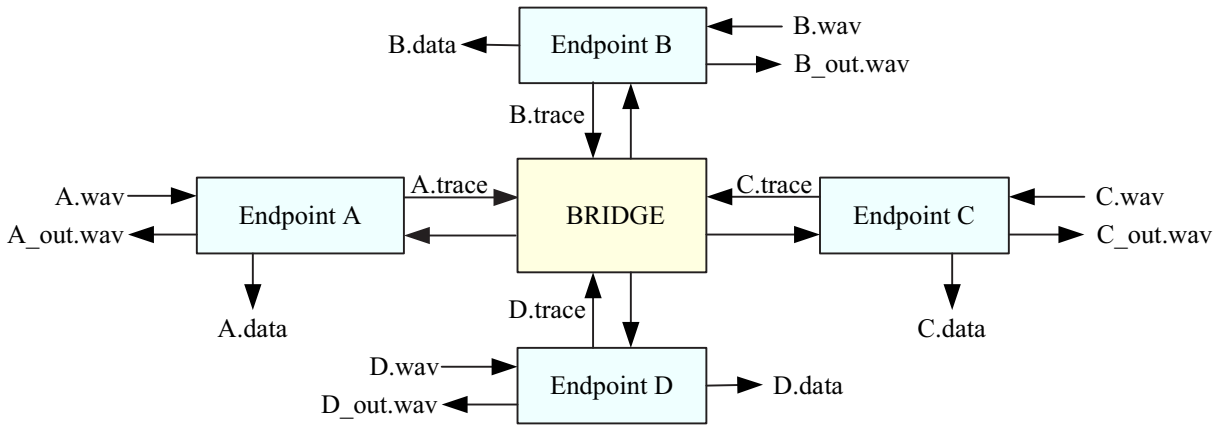


Fig. 4.1 Conference Simulator.

[1]Conference input audio was recorded in a zero delay environment, so the effect of delay on conferee's response times is not apparent in the input audio.

Simulator endpoints are configurable in terms of packetization interval, playout algorithm and playout algorithm parameters (e.g., target loss rate, spike detection, spike detection thresholds, histogram length, etc.). Simulator bridges can be of the Select-and-Forward or Synchronized variety. The Synchronized bridge can bundle selected packets or send them separately. Bridge intrastream synchronization algorithms are configurable in terms of the number of delay values kept in the histogram, and the target late rate.

Data output by the simulator allows for the calculation of average end-to-end delay experienced by selected packets, and packet loss rates at each endpoint and for the conference as a whole. A voice synchronization measure is obtained by taking the average difference in generation time of samples played out at the same time at destination endpoints. In addition, the speaker selection error rate can be evaluated. This is done by comparing the packets that were selected as speakers as compared to the optimum speaker selection decisions calculated off-line in a zero delay environment. MATLAB routines are used to process statistical output from the simulator and create graphical delay plots.

The conference simulator also allows for the simulation of sampling clock skew. By making the sampling clock of endpoint A advance once every X ticks of the global event clock and that of endpoint B advance once every $X + 1$ ticks of the global event clock, a relative skew of $\frac{X}{X+1}$ can be simulated. Input audio for the *skewed* endpoint needs to be adjusted by dropping or repeating silent samples as the rate at which samples are packetized will be different from the nominal sampling rate of the audio file.

4.2 Experimental Method

Conference simulations were done based on audio from an eight minute, four person conference, taken from [5]. Four sets of network delay configurations were used, with network delay traces selected from [82]. Simulations used two selected speakers to represent the conference (i.e., $M = 2$) and an RTP packetization interval of 20ms.

Two bridge types were evaluated: the Select-and-Forward bridge presented in Section 2.7.1 and the Synchronized bridge presented in Chapter 3. Target late rates were varied from 5% to 50% for the Synchronized bridge.

In order to get a network delay vs. packet loss curve for each bridge, simulations were repeated for endpoint target loss rates ranging from 1% to 25% (in increments of 2%) for each bridge type. In addition, bridges were evaluated using three different types of endpoint

playout algorithms. The first algorithm is a histogram-based fixed playout algorithm with spike detection that allows adaptation by dropping or repeating silence packets, based on [44], the second is Ramjee's talkspurt adaptive algorithm 4 [41] and the third is based on Liang's packet-adaptive algorithm [52], which uses WSOLA for time-scaling of audio packets.

Table 4.1 Delay trace configurations used for simulations. Traces and data were taken from [82], except for the last two, which were generated randomly with delays uniformly distributed between 1 and 10ms.

Trace	Min (ms)	Max (ms)	Mean (ms)	Std. Dev. (ms)
Trace Set 1				
OttNight.log	40	236	49.4	8.5
TOday.log	21	526	33.0	11.6
TO2day.log	20	285	32.1	10.1
TO2night.log	20	1387	31.1	15.0
Trace Set 2				
YorkNight.log	10	991	13.0	6.4
UTday.log	9	2110	27.6	59.4
UT2day.log	10	1035	21.9	52.9
UT2night.log	9	556	12.4	15.0
Trace Set 3				
YorkNight.log	10	991	13.0	6.4
UTnight.log	9	662	16.3	32.7
UBCday.log	67	569	72.2	20.4
UBCnight.log	67	727	78.5	33.7
Trace Set 4				
UBCday.log	67	569	72.2	20.4
UBCnight.log	67	727	78.5	33.7
rand1_10A.log	1	10	5.48	2.87
rand1_10B.log	1	10	5.48	2.87

The four sets of delay traces used in simulations, each one representing a different distribution of network delay patterns across conference participants, are shown in Table 4.1.

Traces were taken based on 10ms packets, so every second delay value from the trace was taken when simulating 20ms packets (and every fourth for 40ms packets). For Trace Set 1, all conferees experience similar low-jitter network characteristics, while Trace Set 2 still provides similar network delays across all conferees, but with much larger delay jitter on two of the traces (UTday.log and UT2day.log). Trace set 3 uses two relatively low delay and two high delay traces, while Trace Set 4 simulates a scenario where two participants experience little delay and jitter on their source-to-bridge path, while the other two conferences experience much higher and more variable delays. Each trace represents the delay characteristics for the endpoint-to-bridge path as well as the return bridge-to-endpoint path.

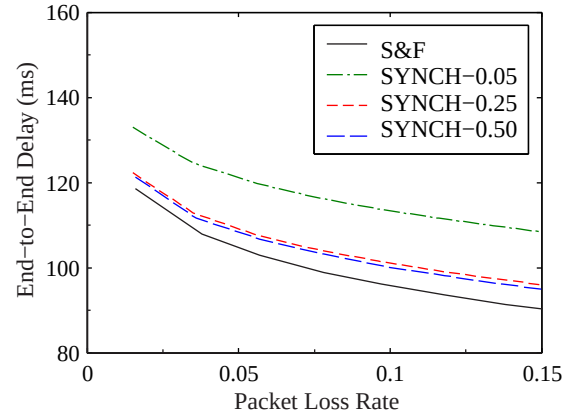
4.3 Delay and Packet Loss

End-to-end delay and packet loss rate are two key metrics in evaluating any VoIP system. The end-to-end delay here is defined as the *time between a packet leaving the original source and being played out at the final destination.* Each conference participant will thus see a different end-to-end delay on a given selected packet, due to differences in network delay between the bridge and each endpoint. A *lost* packet is defined as any packet that does not arrive before its scheduled playout time at the final destination[2]. In addition to lost packets, any additional gaps in playout are counted as part of the packet loss rate. Consider a scenario where packet i arrives late for its playout time t_i, and packet $i+1$ is then scheduled for playout at $t_{i+1} > t_i + T$. Even if packet $i+1$ arrives in time for its scheduled playout, the gap between the playout of packet $i-1$ and packet $i+1$ will be greater than one packet period (T). This scenario will thus be counted as $\frac{t_{i+1}-t_{i-1}}{T}$ lost packets[3].

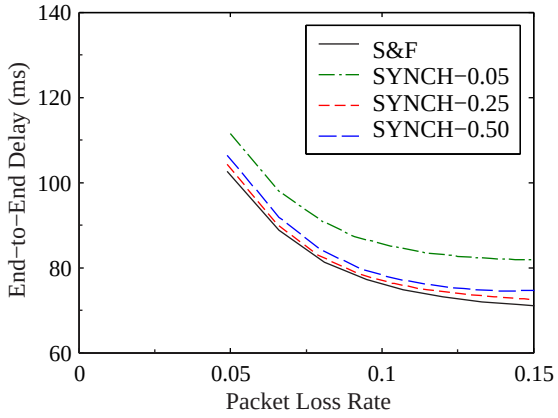
Fig. 4.2 presents end-to-end delay vs. loss curves for each of the four trace sets where endpoints use a histogram-based fixed playout algorithm, while Fig. 4.3 presents the same curves but with endpoints using a packet-adaptive algorithm where time-scale modification of packets is performed at endpoints. Table 4.2 and Table 4.3 present the same data in table format by showing average end-to-end delays for all selected packets at a packet loss rate of 5%, along with the average buffering delay and delay abstraction error experienced at

[2]This includes packets that never arrive.

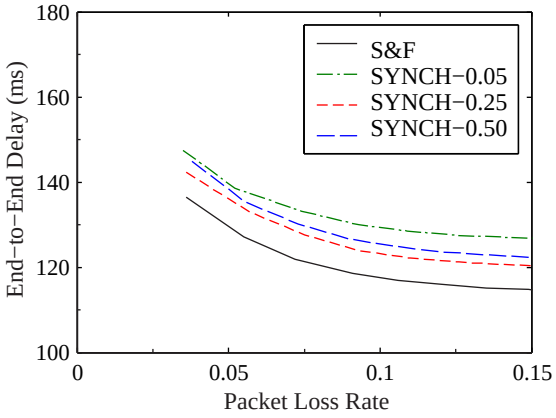
[3]This occurs in many algorithms with a spike detection component, such as in [44].



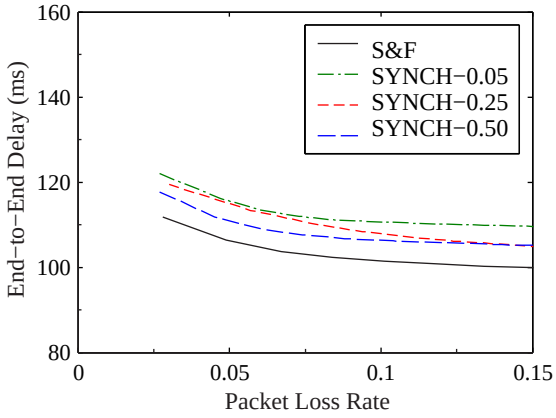
(a) Network Trace Set 1.



(b) Network Trace Set 2.

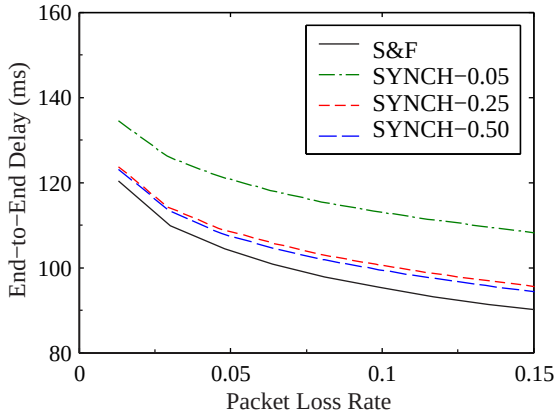


(c) Network Trace Set 3.

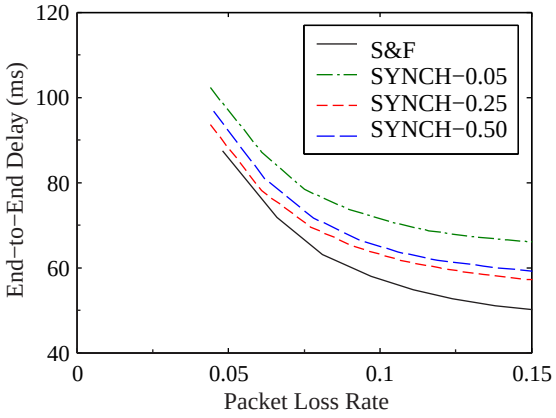


(d) Network Trace Set 4.

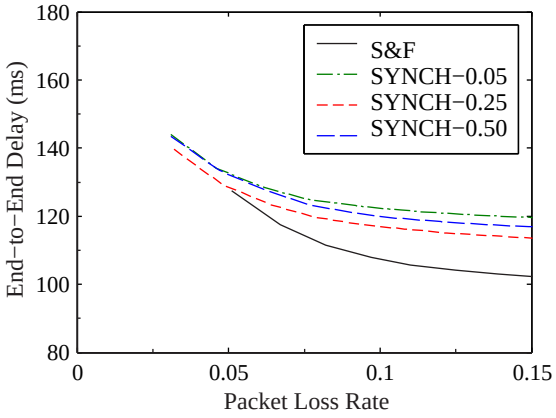
Fig. 4.2 Delay vs. Packet Loss for conferences with endpoints using fixed playout algorithms. S&F is the Select-and-Forward bridge while Synch-X represents a synchronized bridge with target late rate of X.



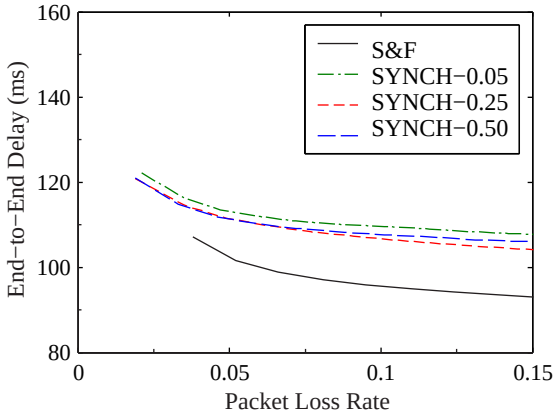
(a) Network Trace Set 1.



(b) Network Trace Set 2.



(c) Network Trace Set 3.



(d) Network Trace Set 4.

Fig. 4.3 Delay vs. Packet Loss for conferences with endpoints using packet-adaptive playout algorithms. S&F is the Select-and-Forward bridge while Synch-X represents a synchronized bridge with target late rate of X.

the bridge. Simulations with endpoints using Ramjee's algorithm 4 resulted in significantly poorer performance than those using fixed or packet-adaptive playout algorithms and are not shown here.

Table 4.2 Delay at 5% packet loss for conferences using Trace Sets 1 and 2. E-E(F) is the average end-to-end delay over all selected packets with endpoints using a fixed playout algorithm while E-E(P) is with endpoints using a packet-adaptive playout algorithm. BB represents the average amount of time that selected packets were buffered at the bridge, while AE represents the average delay abstraction error experienced by all selected packets. Values in brackets for the Synchronized bridge represent the target late rate at the bridge.

Bridge	Trace Set 1				Trace Set 2			
	E-E(F) (ms)	E-E(P) (ms)	BB (ms)	AE (ms)	E-E(F) (ms)	E-E(P) (ms)	BB (ms)	AE (ms)
S&F	104.8	104.0	0	n/a	101.9	85.8	0	n/a
Synch(0.05)	121.2	120.7	23.0	0.4	111.5	97.0	12.3	5.7
Synch(0.15)	111.1	110.3	11.1	1.7	104.2	90.7	2.9	7.1
Synch(0.25)	109.2	108.5	8.7	2.2	103.4	88.2	1.4	7.9
Synch(0.50)	108.3	107.4	6.2	3.3	105.6	92.1	4.3	6.9

Table 4.3 Delay at 5% packet loss for conferences for Trace Sets 3 and 4. Abbreviations used are the same as in Table 4.2.

Bridge	Trace Set 3				Trace Set 4			
	E-E(F) (ms)	E-E(P) (ms)	BB (ms)	AE (ms)	E-E(F) (ms)	E-E(P) (ms)	BB (ms)	AE (ms)
S&F	129.6	127.5	0	n/a	106.2	102.4	0	n/a
Synch(0.05)	139.7	132.6	9.2	2.3	115.6	113.2	10.8	1.0
Synch(0.15)	136.7	128.4	3.2	3.2	115.1	111.9	7.3	2.0
Synch(0.25)	136.1	128.1	2.4	3.3	114.9	111.4	5.8	2.8
Synch(0.50)	138.3	132.3	6.3	2.6	110.9	111.3	6.9	2.2

As can be seen by the results, the Synchronized bridge results in additional end-to-end delay as compared to the Select-and-Forward case, especially at the low target late rate of 5%. It can also be seen that the delay penalty was, on average, more when using packet-adaptive playout algorithms at endpoints as opposed to a fixed playout algorithm. Raising the target late rate at the bridge allows for a reduction in this delay penalty.

In some instances, target late rates of 50% at the bridge yielded higher average buffering delays at the bridge than late rates of 25% or even 15% (see Trace Sets 2, 3 and 4). While the intrastream synchronization stage will introduce less buffering in the 50% target late rate case than in the 25% or 15% target late rate cases, the interstream synchronization stage is not as easily controlled and may introduce a higher buffering delay even if the target late rate is higher. This can occur when the *master* stream has a greater reduction in intrastream buffering than a slave stream for a common increase in target late rate. Fig. 4.4 illustrates such a scenario.

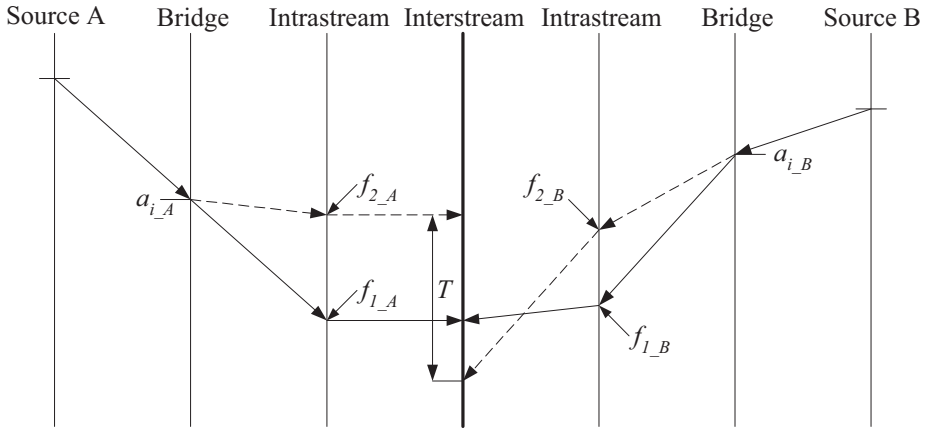


Fig. 4.4 Increase in Buffering Delay despite lower Target Late Rate. Here the stream from source A is the master and that from source B the slave. f_{1_A} and f_{1_B} are the forwarding times after the intrastream synchronization stage for target late rate l_1, where $f_{1\_B} < f_{1\_A}$, and stream B is interstreamed synched to f_{1_A}. For a second, higher, target late rate, l_2, the intrastream forwarding times for streams A and B are f_{2_A} and f_{2_B}, this time where $f_{2\_A} < f_{2\_B}$, and stream B is interstream synched to $f_{2\_A} + T > f_{1\_A}$. Stream B thus experiences a higher buffering delay despite a higher target late rate.

Trace Set 1 provided the highest buffering delays at the bridge, despite being comprised of traces with relatively low network delay variances. This can be attributed to the low frequency of network delay spikes as compared to the other Trace Sets. Most of the higher network delay variances found in traces used in sets 2, 3 and 4 can be attributed to large delay spikes, which are ignored in the intrastream synchronization stage at the bridge (see Section 3.1.1).

4.4 Speaker Selection Error Rate

An *optimum* set of speaker selection decisions for a given conference can be established by applying the speaker selection algorithm offline to audio files, simulating a zero delay environment. When these audio files are packetized and each subjected to different and varying delays, speaker selection decisions will deviate from this optimum.

The *Speaker Selection Error Rate* for a given conference simulation can be measured by taking the percentage of packets that were incorrectly (or sub-optimally) selected as part of the speaker's stream. Erroneous speaker selections comprise both packets that were selected as speakers that were not selected as speakers in the optimal case, and packets that were selected as speakers in the optimal case that were not selected as speakers in the simulation in question.

An additional metric of *Average Forwarding Error* is used in tandem with the Speaker Selection Error Rate to give a better evaluation of the packets that are forwarded from the bridge. The Forwarding Error for a given packet period (in this case for a 20ms period) is defined as the difference between the number of packets that were forwarded as compared to the number that should have been forwarded (M — in this case 2). The Average Forwarding Error is then the average of forwarding errors over all packet periods. The Forwarding Error is presented as a complementary metric because the Select-And-Forward bridge has no restrictions on the number of packets that can be forwarded for a given packet period — a packet due to be forwarded will rarely not be forwarded, but packets are often forwarded when they would not have been in the optimal case.

Table 4.4 Speaker Selection Error Rate. SSE is the speaker selection error rate (in %) while FE is the average forwarding error (in number of packets).

Bridge	Trace Set 1		Trace Set 2		Trace Set 3		Trace Set 4	
	SSE	FE	SSE	FE	SSE	FE	SSE	FE
S&F	1.93	0.37	4.94	0.31	5.96	0.16	5.32	0.10
Synch(0.05)	4.04	0.09	4.40	0.22	6.89	0.09	7.13	0.03
Synch(0.15)	4.23	0.06	4.14	0.25	6.69	0.14	7.23	0.05
Synch(0.25)	4.26	0.37	4.06	0.27	6.65	0.15	7.24	0.05
Synch(0.50)	4.99	0.41	4.22	0.23	6.75	0.30	6.73	0.03

As can be seen in Table 4.4, the Select-And-Forward bridge generally provides a lower

Speaker Selection Error, while the Synchronized bridge provides a lower forwarding error. The higher forwarding error for the Synchronized bridge can be partially attributed to the restriction that only M (in this case 2) packets can be forwarded for a given packet period[4]. If a packet is erroneously selected, then another packet that should have been selected will be not selected; in essence, each wrongly selected packet will cause two speaker selection errors. The Select-and-Forward bridge has no such restrictions as it does not perform any stream or header translation.

Tightly synchronized bridges (those with low target late rates) have a much lower average forwarding error. Forwarding errors will still occur when selected packets arrive late for forwarding at the bridge. This will occur more often as the target late rate increases.

Overall, the Synchronized bridge does not improve speaker selection accuracy over the Select-and-Forward case, but does provide a more periodic forwarding of selected packets, especially at lower target late rates.

4.5 Voice Synchronization

Perfect voice synchronization occurs when the voice samples making up the composite signal representing the conference have the exact same generation time. The level of voice synchronization can be measured by taking the average difference in generation time of voice samples played out at the same time at endpoints.

For conference simulations, a *listening* endpoint is used in addition to the four active endpoints. Because other endpoints will not receive forwarded packets for which they were the original source, only the listening endpoint will receive all selected packets, since it is never selected as speaker. As an additional synchronization metric, the percentage of sample periods for which exactly M (in this case 2) samples are mixed together is used.

Table 4.5 presents voice synchronization results. The Synchronized bridge does not improve voice synchronization as compared to the Select-and-Forward bridge. This can be attributed to the fact that the interstream synchronization stage at the bridge only attempts to line up packet boundaries, without explicitly doing any *group synchronization*.

The synchronization process at the bridge does improve the percentage of sample periods in which the correct number of samples are mixed to form the composite signals at

[4]This is due to the multiplexing of N input streams to M output streams and adherence to the RTP standard for timing of outgoing packets.

Table 4.5 Voice Synchronization. DGT (Difference in Generation Time) is the average difference (in ms) in generation time of samples played out at endpoints. SC (Sources Correct) is the percentage of sampling periods at the listening endpoint that have the correct number (2) of sources being played simultaneously .

Bridge	Trace Set 1		Trace Set 2		Trace Set 3		Trace Set 4	
	DGT	SC	DGT	SC	DGT	SC	DGT	SC
S&F	16.6	94.5	69.4	91.5	63.5	94.3	71.5	95.5
Synch(0.05)	17.1	96.6	68.6	94.7	74.2	96.7	77.5	97.1
Synch(0.15)	14.5	96.6	66.8	93.9	71.6	96.6	71.9	97.5
Synch(0.25)	14.0	96.5	66.6	93.9	70.1	96.6	70.5	97.6
Synch(0.50)	16.5	96.0	66.6	94.1	70.2	96.2	66.2	96.4

endpoints. This percentage generally improves as the target late rate at the bridge is made smaller. This is due to the near-periodic output of packets from the synchronized bridge[5] resulting in M packets leaving the bridge at the same time for most packet periods.

4.6 Speech Quality

Speech quality was measured subjectively by listening to output audio of the *listening* endpoints for simulations with the Select-and-Forward bridge and Synchronized bridge. Output audio was taken from endpoints using a fixed playout algorithm with target late rate of 1%, so as to minimize speech degradation based on packet loss at endpoints.

For each trace set, listeners were presented with four audio files and asked to rank them. One file was simply the summation of the input audio (i.e., the optimal conference), another was the simulation output for the Select-and-Forward bridge, while the other two were simulation outputs for the Synchronized bridge at target late rates of 5% and 25%. Since there were 16 audio files (4 for each of the 4 trace sets), each 8 minutes in length, several specific sections of the audio were targeted, such that the listeners did not have to listen to audio output in their entirety. The selected segments all had a high level of interaction between conference participants, where 3 or more people were talking back in forth in quick succession, or all at the same time.

The Synchronized bridge was judged to have the lowest quality of speech during these

[5]This output will be more and more periodic as the target late rate at the bridge is made smaller.

highly interactive segments, with the Synchronized bridge with a target late rate of 25% performing worse than that with a 5% target late rate. This was particularly evident for output audio from simulations using Trace Set 2. Output audio from Trace Sets 1 and 4 showed little difference among output audio files, while output audio from the Select-and-Forward bridge simulation using Trace Set 3 did result in better speech quality for approximately one third of the selected segments.

The poorer performance of the Synchronized bridge during periods of high interactivity can be attributed to the restriction of forwarding no more than M packets for a given packet period. If a stream has packets arriving late for its scheduled forwarding time at the bridge, its promotion to a selected speaker can be delayed by one or more packets.

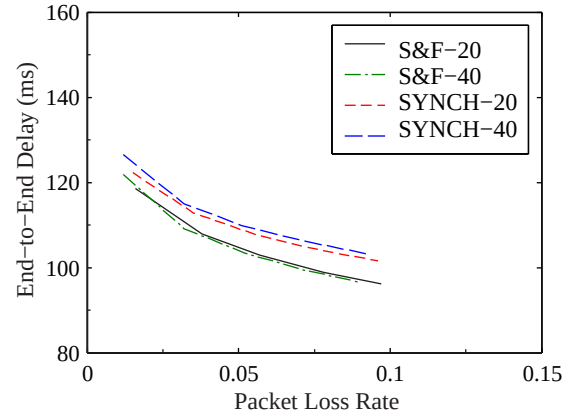
Consider the case where sources A and B are the currently selected packets, and packets from sources A, B, and C arrive at the bridge at times t_A, t_B, and t_C. These packets are all assigned the same synchronized forwarding time, s_i, and $t_A < t_B < s_i < t_C$ (the packet from source C arrives late for its scheduled forwarding time). If the packet from source C has signal power, E_C, large enough such that the updated power signal envelope of source C would promote it to a higher speaker rank than source A and/or B, the packet will still not be forwarded. Sources A and B will be selected as speakers for the period defined by s_i and their packets will be forwarded. Since M packets will have already been forward for forwarding time s_i when the packet from source C arrives, the packet from source C will be discarded. The same scenario in the Select-and-Forward case would results in all three packets being forwarded.

During periods where there was not a very high level of interactivity between conference participants, there was no noticeable difference between the output audio from the Select-and-Forward bridge simulations and that from either of the Synchronized bridge simulations.

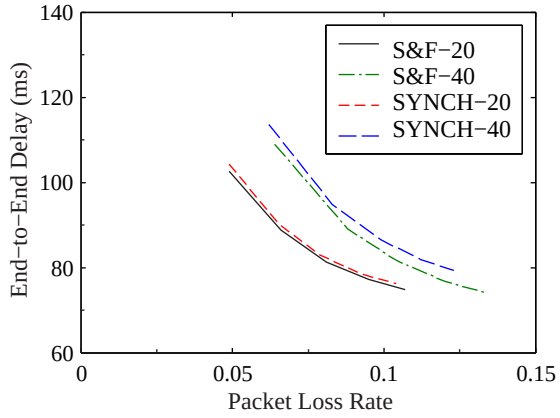
4.7 Effect of Packet Size

For the Synchronized bridge, a larger packet size will decrease the adaptability of the intrastream synchronization stage and potentially increase the delay caused by interstream synchronization.

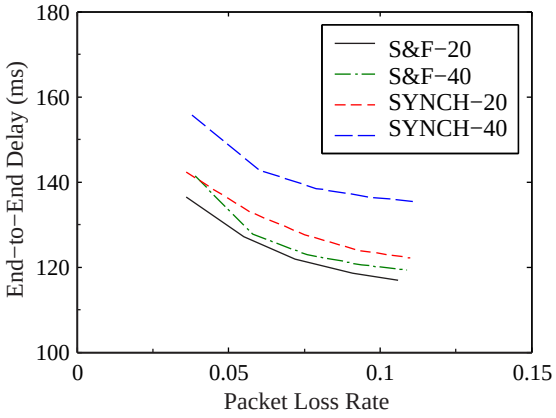
Simulations were repeated for both the Select-and-Forward bridge and the Synchronized bridge using a packet length of 40ms instead of 20ms. Delay-loss curves are shown in Fig. 4.5



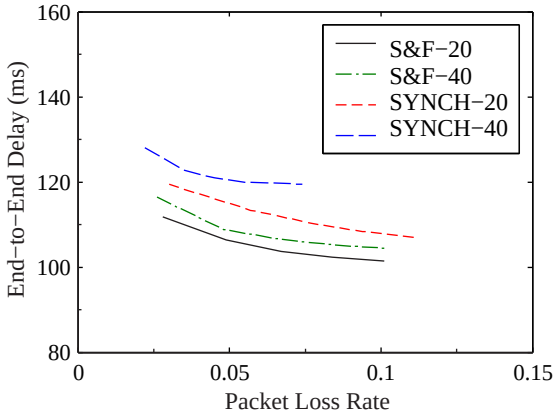
(a) Network Trace Set 1.



(b) Network Trace Set 2.



(c) Network Trace Set 3.



(d) Network Trace Set 4.

Fig. 4.5 Delay vs. Packet Loss for conferences with endpoints using fixed playout algorithms and 40ms packets. A target late rate of 25% is used for the Synchronized bridge. Curves for corresponding 20ms simulations are also shown. Numbers specified after the bridge type in the legend refer to the packet size.

and the corresponding data is shown in Table 4.6.

Table 4.6 Delay at 5% packet loss for conferences with 20ms and 40ms packets. A target late rate of 25% was used for the synchronized bridge and a fixed playout algorithm was used at endpoints.

Bridge	Trace Set 1		Trace Set 2		Trace Set 3		Trace Set 4	
	E-E (ms)	BB (ms)	E-E (ms)	BB (ms)	E-E (ms)	BB (ms)	E-E (ms)	BB (ms)
S&F-20ms	104.8	0	101.9	0	129.6	0	106.2	0
S&F-40ms	104.0	0	120.3	0	133.5	0	108.6	0
Synch-20ms	109.2	8.7	103.4	1.4	136.1	2.4	114.9	5.8
Synch-40ms	110.1	9.5	124	2.4	148.7	11.5	120.5	15.3

An increase in packet size affects both the bridge's ability to adapt its forwarding time as well as limiting the ability of endpoints to adjust their playout time, especially when a fixed playout algorithm is used. End-to-end delays increased both for Select-and-Forward and Synchronized bridge conferences. In general, the delay penalty attributable to increased packet size is slightly greater for Synchronized bridge simulations.

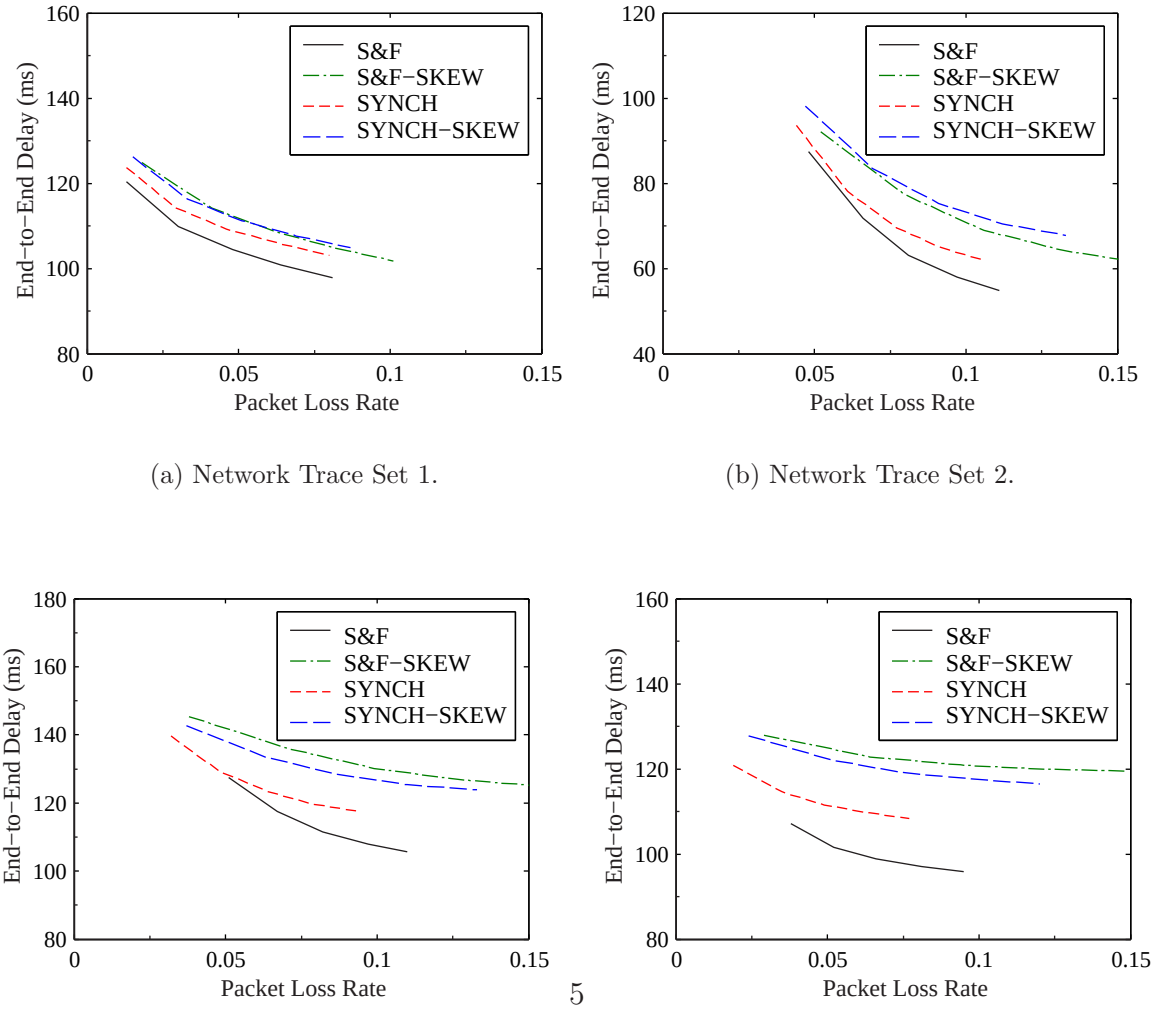
Simulations done for endpoints using packet-adaptive playout algorithms show negligible differences in delay in Synchronized bridge simulations for Trace Sets 1 and 2, and small increases in end-to-end delay (on the order of 5ms) for Trace Sets 3 and 4.

4.8 Effect of Clock Skew

Simulations were run under skew conditions for both the Synchronized bridge and the Select-and-Forward bridge. In order to simulate skew, the actual sampling clock rate of one endpoint was set to 0.5% greater than the nominal rate of 8 kHz, while another was set to 0.5% slower[6]. The remaining endpoints and the bridge had sampling clock rates equivalent to the nominal rate.

Delay-loss curves for skew simulations with endpoints using a packet-adaptive playout algorithm are shown in Fig. 4.6 for the Select-and-Forward bridge and a Synchronized bridge with target late rate of 25%. Curves for the non-skew simulations under the same

[6]Due to the granularity of the simulator event clock, the actual sampling rates of skewed endpoints were 7962 kHz and 8039 kHz.



(a) Network Trace Set 1.

(b) Network Trace Set 2.

(c) Network Trace Set 3.

(d) Network Trace Set 4.

Fig. 4.6 Delay vs. Packet Loss for conferences with endpoints using packet-adaptive playout algorithms under skew conditions. A target late rate of 25% is used for the Synchronized bridge. Curves for corresponding non-skew simulations are also shown.

network conditions are shown for comparison. Table 4.7 presents the data in table format for the same bridges, as well as a Synchronized bridge with target late rate of 5%.

Table 4.7 Delay at 5% packet loss for conferences under clock skew conditions. A packet-adaptive playout algorithm was used at endpoints for all conferences. Target late rates at the bridge are shown in brackets.

Bridge	Trace Set 1		Trace Set 2		Trace Set 3		Trace Set 4	
	E-E (ms)	BB (ms)	E-E (ms)	BB (ms)	E-E (ms)	BB (ms)	E-E (ms)	BB (ms)
S&F	111.8	0	93.6	0	141.9	0	124.9	0
Synch(0.05)	123.4	23.9	106.6	18.2	142.3	12.5	122.8	13.8
Synch(0.25)	111.5	9.1	96.2	4.8	138.1	6.5	122.4	9.3

Results from simulations with endpoints using fixed playout scheduling algorithms are not included here, as most could not achieve overall packet loss rates of less than 5% under clock skew conditions. While achievable packet loss rates for most simulations were high, the Synchronized bridge strongly outperformed Select-and-Forward bridges in terms of end-to-end delay when using fixed playout scheduling algorithms at endpoints under clock skew conditions.

For conference simulations with endpoints using packet-adaptive playout algorithms, the Synchronized bridge outperforms the Select-and-Forward bridge in terms of end-to-end delay for a given loss rate, despite increased buffering at the bridge as compared to the non-skew case (see Table 4.2 and Table 4.3).

The Synchronized bridge's greater robustness to clock skew as compared to the Select-and-Forward bridge arises from the fact that an endpoint in a Select-and-Forward conference will see potentially long gaps in RTP streams during periods where a given source endpoint is not selected as speaker. While an endpoint in a Select-and-Forward environment will see $N-1$ streams, M of which are active at any given time, and endpoint in a Synchronized environment will see M streams which are always active. Gaps in RTP streams force endpoints to use calculated network delay values from far in the past when scheduling newly arrived packets for playout. When there is clock skew, the offset between sampling clock counters will drift over time. Packet i arriving at time a_i on the destination

sampling clock will have a calculated delay, r_i, of

$$\begin{aligned} r_i &= a_i - f_i \\ &= b_i + t_{off}, \end{aligned} \tag{4.1}$$

where f_i is the timestamp taken from the source sampling clock, b_i is the actual network delay experienced by the packet, and t_{off} is the offset between source and destination sampling clocks[7]. The offset, t_{off}, will change at a rate equal to the sampling clock skew between source and destination. If there is a gap of length N packets in a given stream, t_{off} will change by an amount equal to sNT, where s is the clock skew between source and destination and T is the packet length. If packet i arrives before a gap of length N packets, the *calculated* network delay difference between packet $i+1$ and packet i will be

$$\begin{aligned} r_{i+1} - r_i &= (b_{i+1} + t_{off} + sNT) - (b_i + t_{off}) \\ &= b_{i+1} - b_i + sNT. \end{aligned} \tag{4.2}$$

Even if b_i and b_{i+1} are equal, there will be large *perceived* differences if N is big. Stream gaps will cause playout scheduling errors as the calculated network delays on which playout scheduling for packet $i+1$ is based will have a significantly different sampling clock offset, t_{off}. A scenario with a large gap in the RTP stream is shown in Fig. 4.7.

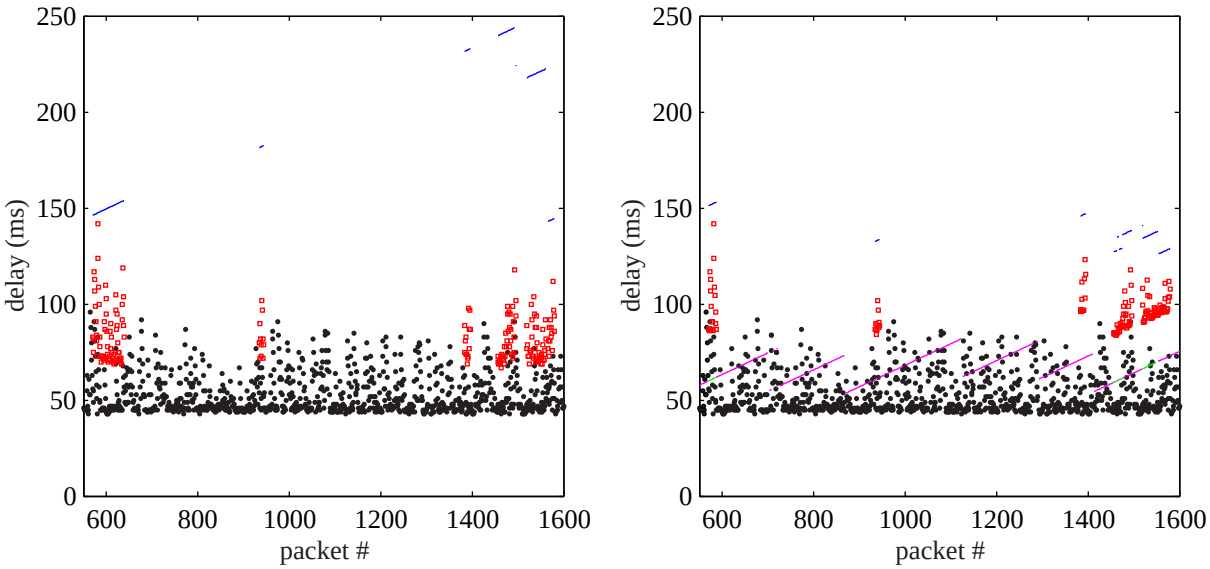
In addition to providing a continuous RTP stream by mapping input streams to *speaker* streams, the Synchronized bridge also provides immunity to clock skew through the intrastream synchronization process, which abstracts the skew by adjusting forwarding times at the bridge, in effect performing a periodic skew correction.

4.9 Implementation Complexity

4.9.1 Bridge Complexity

The Select-and-Forward bridge is less complex than the Synchronized bridge, in that it does not multiplex incoming RTP streams and perform source translation. The Synchronized

[7]This offset is not typically known. The calculated delay, r_i, is used for playout scheduling.



(a) Packets travelling from endpoint A to endpoint B in conference with Select-and-Forward bridge.

(b) Packets travelling from endpoint A to endpoint B in conference with Synchronized bridge.

Fig. 4.7 Effect of Clock Skew. The bottom (black) dots represent the packet arrival time at the bridge (i.e., the source-to-bridge delay). The small (red) squares represent the arrival time at the destination. The top (black) line represents playout at the destination. Spaces with no (red) squares or top line represent periods where endpoint A is not selected as a speaker (no packets are being forwarded). The lower (magenta) line in (b) represents the scheduled forwarding time of packets at the bridge. Lines are diagonal (as opposed to horizontal) due to clock skew between source and destination. The large gap in the RTP stream (from around packet 650 to 1400) results in excessive playout buffering in (a) due to the change in sampling clock offsets between source and destination. In (b), under the same network conditions, the destination does not see a gap in the RTP stream because the packets are being multiplexed by the bridge onto one of the *speaker* streams.

bridge has the added tasks of intrastream and interstream synchronization[8], as well as more involved packet header translation and management of packets that arrive late for forwarding at the bridge.

4.9.2 Endpoint Complexity

Because the Select-and-Forward bridge does not abstract the conference from endpoints, endpoints in a Select-and-Forward conference environment must handle $N-1$ RTP sessions, one with each other endpoint involved in the conference. While only M of these streams will be active at any given time, playout scheduling will need to be done independently on all $N-1$ streams. If any new endpoints join the conference, an additional RTP streams will have to be set up between it and all other endpoints.

An endpoint in a Synchronized bridge conference environment need only support M RTP streams, as the conference details are abstracted by the bridge. The addition of any new endpoints to the conference will be transparent to other endpoints and require no action on their part.

In both the Select-and-Forward and Synchronized environments, endpoints will be required to mix streams.

4.10 Scalability

One of the main advantages of the Synchronized bridge over the Select-and-Forward bridge is its modular nature, which allows a conference to scale nicely to large or multiple bridge conferences. Requirements placed on endpoints in a Synchronized conference environment will not increase with N. The processing load on the bridge will increase with N, but the modular nature of the bridge allows for easy load sharing with the addition of supplemental bridges, as shown in Fig. 4.8.

The abstraction of conference details from endpoints in a Synchronized environment also allows for easy support of dynamic conferencing arrangements, as signalling requirements are only between bridge and endpoints.

The Select-And-Forward conferencing environment scales very poorly to multiple bridge or large conferences. Because the bridge does not abstract the rest of the conference from

[8] In the design presented in Chapter 3, interstream synchronization is only performed once for each input stream, after the initial warm-up period.

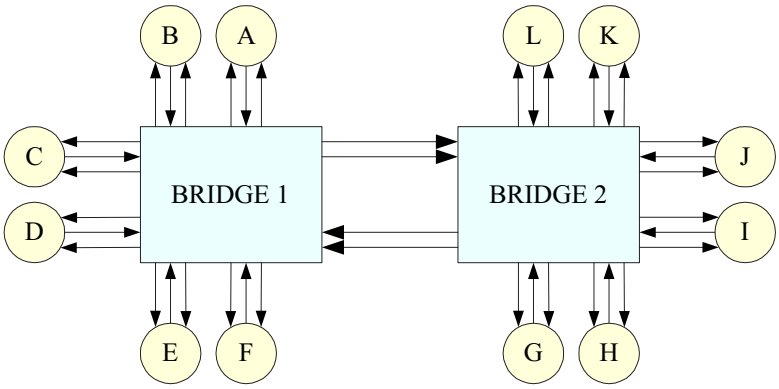


Fig. 4.8 Multiple Bridge Conference with Synchronized Conference Bridge [79]. Here $M = 2$ for both bridges, and $N = 12$. In the same Select-and-Forward multiple bridge configuration, each endpoint would need to support 11 RTP streams.

the endpoints or other bridges, the latter are required to support RTP sessions with all other endpoints involved in the conference. The Select-And-Forward conferencing environment also adapts poorly to new conferees being added to a conference. Endpoints must be involved in any signalling setup upon arrival of new conference participants in order to establish a new RTP session with that new conferee.

Chapter 5

Conclusion

The best-effort, packet-based nature of wide-area IP networks results in network delay variability on the order of multiple packet lengths. The Synchronized bridge presented in this thesis provides a synchronization mechanism for mapping N input voice streams to M output "speaker" streams in such high-jitter environments. The Synchronized bridge acts as an RTP mixer, performing the requisite header translation to ensure timing adherence to the RTP standard for the M output streams. By performing this stream translation, the Synchronized bridge abstracts details of source-to-bridge delay characteristics, providing a modular and scalable conferencing environment suitable for IP networks, while at the same time simplifying signalling requirements for conference endpoints.

5.1 Summary and Discussion of Results

This thesis introduced a design for a Synchronized conference bridge that uses novel synchronization algorithms to allow the bridge to multiplex N input voice streams to M output streams representing selected speakers. This bridge was evaluated using a conference simulator, also developed for this thesis, and performance of the Synchronized bridge was compared to that of a Select-and-Forward bridge, presented in Section 2.7.1.

Protocols, components and algorithms used in VoIP systems were introduced in Chapter 2, with special focus on speech codecs and intrastream and interstream synchronization. The basics of centralized conferencing were also addressed in this chapter, as were details of the Select-and-Forward bridge design, earlier proposed in [5].

Chapter 3 provided the design details of the Synchronized bridge, including those of

novel intrastream and interstream synchronization algorithms. The effects of these synchronization stages on overall delay were examined and discussed, as was the effect of *late* arriving packets. RTP packet header translation required for transparent mapping of streams was detailed. Extensions to bundling were presented, as were requirements on endpoints in a Synchronized conferencing environment. Finally, the implications of using state-based codecs, and the state synchronization errors that can occur during speaker transition, were examined.

Simulation results for the Synchronized and Select-and-Forward bridges were presented in Chapter 4. Evaluations examined delay vs packet loss, voice synchronization, speaker selection accuracy, and a qualitative analysis of simulator output audio to assess speech quality. The effects of packet size and endpoint sampling clock skew were also examined for the two bridge types.

The synchronization of streams at the Synchronized bridge was shown to inject additional end-to-end delay, although this delay penalty was on the order of half of one packet length, or less, for all simulations. Delay penalties were higher on average when conference endpoints used packet-adaptive as opposed to fixed or talkspurt-adaptive playout algorithms, although in most cases only by a few milliseconds.

Conferences using Synchronized bridges showed greater robustness to endpoint clock skew, outperforming the Select-and-Forward environment in terms of end-to-end delay in most simulations under skew conditions.

The synchronization process resulted in additional speaker selection errors on some speaker transitions, due to the restriction of only being able to forward M packets per packet period. These speaker selection errors resulted in noticeable degradation of conference audio quality during periods of high interactivity between conferees.

The Synchronized bridge provides a more modular and scalable conferencing architecture than that provided by the Select-and-Forward bridge. Signalling requirements on endpoints are reduced as they need only support M streams originating from the bridge, instead of $N-1$ streams originating from the other conference endpoints. There is a slight degradation in performance when using a Synchronized bridge, due to additional delay introduced during the synchronization stage, and due to speaker selection errors caused by the adherence to M output streams.

5.2 Future Work

This section covers incomplete research, as well as suggesting possible approaches to improve performance of the Synchronized conference bridge presented in this thesis. Possible extensions to performance evaluations are also considered, as are potential alternate solutions.

5.2.1 Intrastream Synchronization

Adaptive Scheduling of Forwarding Times

The intrastream synchronization process used for the Synchronized bridge is very simple and does not allow for adaptation of forwarding times except by dropping or repeating silence packets. Efforts were made to allow for a more adaptive scheduling of forwarding times by allowing successive packets in a bridge playout buffer to be scheduled for forwarding slightly more or slightly less than one packet period apart. This creates a small deviation from a periodic output of packets from the intrastream stage, but allows for the forwarding time of packets to be slowly adapted by small amounts. Due to the ensuing interstream synchronization stage, this type of adaptation can only be done by the master stream (the stream currently selected as primary speaker) and requires that all slave streams resynchronize to the master stream whenever its forwarding time is adapted. This greatly complicates the interstream synchronization process as compared to the design presented in this work. Better approaches to allowing for a more adaptive intrastream synchronization stage at the bridge should be investigated.

Adaptive Target Late Rate

The best choice of target late rate at the bridge is a function of the network delay characteristics of the endpoint-to-bridge paths, as well as the differences between them. Instead of using a prescribed and fixed target late rate, it could be chosen based on observations of the conference network delay characteristics within and between streams, and be adapted over the course of a conference if the network delay characteristics change.

Spike Detection

The effect of the use of spike detection as part of the bridge intrastream synchronization algorithm requires more investigation. As in the case of choosing a target late rate, the choice of spike thresholds (for determining when to enter or exit spike mode) could be improved by basing them on conference network statistics, and allowing them to be adaptable as delay characteristics change. Different thresholds for different input streams could be used. Improvements to spike detection algorithms could also be applied to playout scheduling at endpoints.

Other scheduling algorithms

The intrastream synchronization algorithm used in the synchronized bridge is histogram-based. Other approaches, such as the use of an auto-regressive estimate, may lessen the computational and memory requirements of the bridge. The impact of using such an algorithm should be investigated.

5.2.2 Interstream Synchronization

Group Synchronization

The interstream synchronization stage only lines up packet boundaries, without taking into account the difference in packet generation times. Group synchronization would improve voice synchronization of mixed streams, which can be important for applications that require fairness. However, group synchronization will likely result in more delay.

Better choice of initial synchronization

The initial interstream synchronization for slave streams has too much bearing on the delay performance of a given conference. Once the initial synchronization is done, all subsequent forwarding times for streams are restricted to being a packet multiple difference of that initial forwarding time.

The initial synchronized forwarding time is based solely on the master stream. It may be more appropriate to use a collaborative approach that optimizes a *synchronization point*, minimizing the combined delay experienced by all streams due to interstream synchronization.

5.2.3 Speaker Selection Accuracy

Adherence to the RTP standard requires that the Synchronized bridge be limited to the number of packets that can be forwarded in a given packet period. This can lead to packets that arrive late at the bridge not being forwarded even though they represent an active speaker (see Section 4.6). This problem could be alleviated by setting up an $(M+1)^{th}$ RTP stream between bridge and endpoints, that could be used only in such scenarios. Alternatively, RTP protocol extensions could provide a solution.

5.2.4 Stream Mapping without Delaying Packets

Experiments were done with a bridge prototype that achieved the mapping of N input streams to M output streams without delaying packets at the bridge. This can be done by performing stream synchronization at bridges as detailed in Chapter 3, but using the synchronized forwarding time only for the purposes of scheduling a periodic speaker selection routine and ranking of selected speakers. Packets arriving from streams that are currently selected as speakers are forwarded immediately from the bridge, regardless of their scheduled forwarding time. Header translation is done as in Chapter 3.

This no-delay synchronization mechanism will not abstract delay jitter on the source-to-bridge path, but will provide some abstraction of the constant portion of the source-to-bridge path delay. This limited delay abstraction can be achieved by the mapping of sequence numbers on input streams to those on output streams[1]. Still, delay abstraction error will increase when packets are not buffered at the bridge.

This approach can also aggravate the delayed speaker transition problem outlined in Section 4.6, as packets are forwarded before their assigned forwarding time, effectively reserving a spot in the outgoing stream for a packet period *in the future*. In essence, forwarding a packet before its forwarding time is equivalent to assuming that it will still be selected as a speaker when that forwarding time arrives.

Bookkeeping required at the bridge becomes more complicated, as ensuring that M packets are forwarded for a given packet period involves keeping track of packets that have already been forwarded as well as keeping track of outstanding (late) packets from selected streams.

[1] Packets leaving sources A and B at the same can be assigned synchronized forwarding times that are multiples of the packet length apart.

Despite its shortcomings, this no-delay approach to synchronization does eliminate buffering delay at the bridge and can be useful when source-to-bridge paths share similar network delay jitter. Solutions to problems outlined in Section 5.2.3 would help mitigate aggravation of the delayed speaker selection problem. Evaluations of this prototype should be investigated more fully.

5.2.5 Codec Issues

While issues arising from codec state synchronization errors were discussed, codecs were not built into the conference simulator. Offline evaluations of the impact of state synchronization errors were carried out for the Select-and-Forward bridge using G.723.1. Integration of codecs into the simulator would allow for easier evaluation of the impact of frame-based codecs on conferencing, especially when stream translation is performed at the bridge. In addition, a method for perceptual evaluation of such an impact would be extremely beneficial.

State Synchronization Errors

The impact of the following types of state synchronization errors should be determined:

- Errors introduced due to gaps in streams caused by dropped packets at the bridge in a Select-and-Forward conference[2].
- The difference between the case where endpoints keep separate decoder state for each of the other conferees and decode based on the CSRC, and where endpoints keep state for each RTP stream that represents a speakers rank (decode based on the SSRC).
- If decoding based on the source identifier (SSRC) for a synchronized bridge, the impact of assigning source identifiers on outgoing packets from the bridge based on speaker rank, as opposed to assigning source identifiers to minimize switches in stream mapping from input to output (see Section 3.3.2).

[2]Informal evaluations were done for this case, showing that little audible distortion was introduced.

Silence Suppression

While the Synchronized bridge is easily compatible with silence suppression schemes, no explicit consideration of Discontinuous Transmission (DTX) or Silence Insertion Descriptor (SID) frames was done in this work.

5.2.6 Bundling and Mixing

Bundling or mixing requires selective dropping of packets that arrive late at the bridge, and/or additional waiting for selected packets to arrive. Investigating algorithms for deciding whether to wait for or drop a late packet for a selected stream, and their consequent effect on conference speech quality, would be useful. The use of packet loss concealment for the generation of signal approximations of contents of late arriving packets for selected streams is also a possible improvement. The impact of target late rate in a bundling/mixing environment should also be investigated, as tighter synchronization (a lower late rate) will be required if the number of packets discarded due to late arrival is to be kept low.

Hybrid Mixing / Tandem-Free Bridges

Prior work in [13, 78] has proposed a bridge that reduces tandem encodings by only mixing when there are two active speakers, instead of always choosing two or more packets for mixing. In periods where there is a lone talker, the selected packet is forwarded without decoding. When using codecs with a look-ahead delay, a discontinuity will occur when there is a transition from a mixed signal to one that is forwarded without decoding (or the other way around), as the decode-mix-encode operation will introduce an additional look-ahead delay. The mixed stream will therefore be delayed as compared to the stream that is simply forwarded. Mechanisms to circumvent this timing issue or mitigate the perceptual effects of switching between mixed and forwarded streams would allow for a hybrid mixing/tandem-free bridge. A possible solution was presented in [83].

5.2.7 Decentralized Conferencing Models

Decentralized models can provide lower delays by eliminating the conference bridge, but require a full-mesh configuration or multicast support and scale poorly as conferences grow large and silence suppression is not used. Conference endpoints could perform their own

speaker selection to decide whether or not to send their current packet, based on signal power of packets received from other endpoints. This distributed speaker selection would minimize delay but may introduce additional speaker selection errors.

5.2.8 Experimental Protocol

Experiments done in this thesis were based on four-person conferences controlled by a single bridge. Test audio for a larger conference would allow for performance evaluation under a multiple bridge environment.

Live experiments with prototype bridges would allow for user feedback on the overall quality of a conference and provide a perceptual evaluation of the combined effects of delay, speaker selection errors, and voice synchronization.

References

[1] E. A. Munter, "Digital conference circuit and method." United States Patent 4,387,457, June 1983.

[2] J. Bellamy, *Digital Telephony*. New York, NY: John Wiley & Sons, Inc., third ed., 2000.

[3] P. K. Edholm, F. Simard, and N. K. Burns, "Packet-based conferencing." Canadian Patent Application 2,422,448, Mar. 2002.

[4] P. J. Smith, P. Kabal, M. Blostein, and R. Rabipour, "Tandem-free VoIP conferencing: A bridge to next generation networks," *IEEE Communications Magazine*, vol. 41, no. 5, pp. 136–145.

[5] P. Smith, "Voice conferencing over IP networks," Master's thesis, McGill University, Montreal, Canada, Jan. 2002.

[6] J. Davidson and J. Peters, *Voice Over IP Fundamentals*. Indianapolis, IN: Cisco Press, 2000.

[7] U. Black, *Voice Over IP*. Upper Saddle River, NJ: Prentice Hall, 2000.

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications." RFC 3550, Internet Engineering Task Force, July 2003.

[9] D. O'Shaughnessy, *Speech Communications: Human and Machine*. New York, NY: IEEE Press, second ed., 2000.

[10] ITU-T Recommendation G.113, "Transmission impairments due to speech processing," Feb. 2001.

[11] ITU-T Recommendation G.107, "The E-model, a computational model for use in transmission planning," July 2002.

[12] ITU-T Recommendation G.114, "One-way Transmission Time," May 2000.

[13] D. Nahumi, "Conferencing arrangement for compressed information signals." United States Patent 5,390,177, Feb. 1995.

[14] ITU-T Recommendation G.711, "Pulse code modulation (PCM) of voice frequencies," Nov. 1988.

[15] ITU-T Recommendation G.726, "40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)," Dec. 1990.

[16] ITU-T Recommendation G.729, "Coding of speech at 8 kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP)," Mar. 1996.

[17] ITU-T Recommendation G.723.1, "Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s," Mar. 1996.

[18] Cisco 1751 Router Sotware Configuration Guide, "Voice over IP overview." [online] http://www.cisco.com/univercd/cc/td/doc/product/access/acs_mod/1700/1751/1751swg/intro.pdf.

[19] P. T. Brady, "A technique for investigating on-off patterns of speech," *Bell System Technical Journal*, vol. 44, pp. 1–22, Jan. 1965.

[20] J. F. Lynch Jr., J. G. Josenhans, and R. E. Crochiere, "Speech/silence segmentation for real-time coding via rule based adaptive endpoint detection," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, (Dallas, Texas), pp. 31.7.1–31.7.4, Apr. 1987.

[21] ITU-T Recommendation P.59, "Artificial Conversational Speech," Mar. 1993.

[22] A. Benyassine, E. Shlomot, H. Y. Su, D. Massaloux, C. Lamblin, and J. P. Petit, "ITU-T recommendation G.729 annex B: A silence suppression scheme for use with G.729 optimized for V.70 digital simultaneous voice and data applications," *IEEE Communications Magazine*, vol. 37, pp. 64–73, Sept. 1997.

[23] N. Jayant, "Effects of packet loss on waveform coded speech," in *Proc. Fifth Int. Conference on Computer Communications*, (Atlanta, GA), pp. 275–280, 1980.

[24] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, pp. 40–48, Sep.–Oct. 1998.

[25] ITU-T Recommendation G.711 Appendix I, "A high quality low-complexity algorithm for packet loss concealment with g.711," Sept. 1999.

[26] E. Mahfuz, "Packet loss concealment for voice transmission over IP networks," Master's thesis, McGill University, Montreal, Canada, Sept. 2001.

[27] C. Montminy and T. Aboulnasr, "Improving the performance of ITU-T G.729A for VoIP," in *Proc. IEEE Int. Conf. on Multimedia and Expo*, (New York, NY), pp. 433–436, July 2000.

[28] P. Gournay, F. Rousseau, and R. Lefebvre, "Improved packet loss recovery using late frames for prediction-based speech coders," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 1, (Hong Kong), pp. 108–111, Apr. 2003.

[29] H. Sanneck and N. Le, "Speech property-based FEC for internet telephony applications," in *Proc. of the SPIE/ACM SIGMM Multimedia Computing and Network Conference (MMCN)*, (San Jose, CA), pp. 38–51, Jan. 2000.

[30] J. G. Gruber and N. H. Le, "Performance requirements for integrated voice/data networks," *IEEE J. Selected Areas Communications*, vol. SAC-1, pp. 981–1005, Dec. 1983.

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

[32] A. Watson and M. A. Sasse, "Measuring perceived quality of speech and video in multimedia conferencing applications," in *Proc. ACM Multimedia*, (Bristol, UK), pp. 55–60, Sept. 1998.

[33] O. Hodson, C. Perkins, and V. Hardman, "Skew detection and compensation for Internet audio applications," in *Proc. IEEE Int. Conf. on Multimedia and Expo*, vol. 3, (New York, NY), pp. 1687–1690, July 2000.

[34] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proc. of the Conf. on Computer Communications (IEEE-Infocom)*, (New York, New York), pp. 227–234, Mar. 1999.

[35] C. Demichelis and P. Chimento, "IP packet delay variation metric for IP performance metrics (IPPM)." RFC 3393, Internet Engineering Task Force, 2002.

[36] N. Laoutaris and I. Stavrakakis, "Intrastream synchronization for continuous media streams: A survey of playout schedulers," *IEEE Network*, vol. 16, pp. 30–40, May 2002.

[37] G. Barberis and D. Pazzaglia, "Analysis and optimal design of a packet-voice receiver," *IEEE Trans. Communications*, vol. COM-28, pp. 217–227, Feb. 1980.

[38] D. Cohen, "Issues in transnet packetized voice communications," in *Proc. Fifth Data Communications Symposium*, (Snowbird, USA), pp. 6.10–6.13, Sept. 1977.

[39] E. Biersack, W. Geyer, and C. Bernhardt, "Intra and interstream synchronization for stored multimedia streams," in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, (Hiroshima, Japan), pp. 372–381, June 1996.

[40] D. Cohen, "Specifications for the network voice protocol (NVP)." RFC 741, Internet Engineering Task Force, Dec. 1976.

[41] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide area networks," in *Proc. of the Conf. on Computer Communications (IEEE-Infocom)*, vol. 2, (Toronto, Canada), pp. 680–688, June 1994.

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

[43] W. A. Montgomery, "Techniques for packet voice synchronization," *IEEE J. Selected Areas Communications*, vol. SAC-1, pp. 1022–1028, Dec. 1983.

[44] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *ACM/Springer Multimedia Systems*, vol. 6, pp. 17–28, Jan. 1998.

[45] J. Pinto and K. J. Christensen, "An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods," in *Proc. IEEE Conf. Local Computer Networks*, (Lowell, MA), pp. 224–229, Oct. 1999.

[46] H. Schulzrinne, "Voice communications across the Internet: a network voice terminal," tech. rep., Dept. Computer Science, U. Massachussets, Amherst, MA, July 1992.

[47] J. Bolot, "End-to-end packet delay and loss behaviour in the Internet," in *Proc. ACM SIGCOMM*, (San Francisco, CA), pp. 289–298, Sept. 1993.

[48] D. Sanghi, A. Agrawala, O. Gudmundsson, and B. Jain, "Experimental assessment of end-to-end behaviour in the Internet," in *Proc. of the Conf. on Computer Communications (IEEE-Infocom)*, vol. 2, (San Francisco, CA), pp. 867–874, Mar. 1993.

[49] C. J. Sreenan, J.-C. Chen, P. Agrawal, and B. Narendran, "Delay reduction techniques for playout buffering," *IEEE Trans. on Multimedia*, vol. 2, pp. 100–112, June 2000.

[50] A. Shallwani and P. Kabal, "An adaptive playout algorithm with delay spike detection for real-time VoIP," in *Proc. IEEE Canadian Conf. Elec. Comp. Eng.*, vol. 2, (Montreal, Canada), pp. 997–1000.

[51] P. DeLeon and C. J. Sreenan, "An adaptive predictor for media playout buffering," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 6, (Phoenix, AZ), pp. 3097–3100, Mar. 1999.

[52] Y. J. Liang, N. Färber, and B. Girod, "Adaptive playout scheduling using time-scale modification in packet voice communications," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 3, (Salt Lake, UT), pp. 1445–1448, May 2001.

[53] F. Liu, J. W. Kim, and C.-C. J. Kuo, "Adaptive delay concealment for internet voice applications with packet-based time-scale modification," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 3, (Salt Lake, UT), pp. 1461–1464, May 2001.

[54] S. Agnihotri, K. Aravindhan, H. Jamadagni, and B. Pawate, "A new technique for improving quality of speech in VoIP using time-scale modification," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 2, (Orlando, FL), pp. 2085–2088, may 2002.

[55] A. K. Anandakumar, A. McCree, and E. Paksoy, "An adaptive voice playout method for VoP applications," in *Proc. IEEE GLOBECOM*, vol. 3, (San Antonio, TX), pp. 1637–1640, Nov. 2001.

[56] Y. Liang, N. Färber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communication over ip networks," *IEEE Trans. on Multimedia*, vol. 5, pp. 532–543, Dec. 2003.

[57] L. Sun and E. C. Ifeachor, "Prediction of perceived conversational speech quality and effects of playout buffer algorithms," in *Proc. IEEE Int. Conf. on Communications*, 2003.

[58] K. Fujimoto, S. Ata, and M. Murata, "Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications," in *Proc. IEEE GLOBECOM*, vol. 3, (Taipei, Taiwan), pp. 2451–2457, Nov. 2002.

[59] J. Rosenberg, L. Qiu, and H. Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet," in *Proc. of the Conf. on Computer Communications (IEEE-Infocom)*, vol. 3, (Tel Aviv, Israel), pp. 1705–1714, Mar. 2000.

[60] C. Boutremans and J.-Y. Le Boudec, "Adaptive joint playout buffer and FEC adjustment for internet telephony," in *Proc. of the Conf. on Computer Communications (IEEE-Infocom)*, vol. 1, (San Francisco, CA), pp. 652–662, Apr. 2003.

[61] H. Melvin and L. Murphy, "An evaluation of the potential of synchronized time to improve voice over IP quality," in *Proc. IEEE Int. Conf. on Communications*, vol. 3, (Anchorage, AK), pp. 1922–1926, May 2003.

[62] R. Steinmetz, "Human perception of jitter and media synchronization," *IEEE J. Selected Areas Communications*, vol. 14, pp. 61–72, Jan. 1996.

[63] C. Liu, Y. Xie, M. J. Lee, and T. N. Saadawi, "Multipoint multimedia teleconference system with adaptive synchronization," *IEEE J. Selected Areas Communications*, vol. 14, pp. 1422–1435, Sept. 1996.

[64] I. F. Akyildiz and W. Yen, "Multimedia group synchronization protocols for integrated services networks," *IEEE J. Selected Areas Communications*, vol. 14, pp. 162–173, Jan. 1996.

[65] J. Escobar, C. Partridge, and D. Deutsch, "Flow synchronization protocol," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 111–121, Apr. 1994.

[66] K. Rothermel and T. Helbig, "An adaptive protocol for synchronizing media streams," *ACM/Springer Multimedia Systems*, vol. 5, pp. 324–336, Sept. 1997.

[67] Y. Ishibashi and S. Tasaka, "A group synchronization mechanism for live media in multicast communications," in *Proc. IEEE GLOBECOM*, (Phoenix, AZ), pp. 746–752, Nov.

[68] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proc. of the Int. Workshop on Network and Operating System Support for Digital Audio and Video*, (Miami, Florida), pp. 12–29, May 2002.

[69] F. Panzieri and M. Rocetti, "Synchronization support and group-membership services for reliable distributed multimedia applications," *ACM/Springer Multimedia Systems*, vol. 5, pp. 1–22, Sept. 1997.

[70] P. V. Rangan, H. M. Vin, and S. Ramanathan, "Communication architectures and algorithms for media mixing in multimedia conferences," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 20–30, Feb. 1993.

[71] Y. Ishibashi and S. Tasaka, "A synchronization mechanism for continuous media in multimedia communications," in *Proc. of the Conf. on Computer Communications (IEEE-Infocom)*, vol. 3, (Boston, USA), pp. 1010–1019, Apr. 1995.

[72] ITU-T Recommendation G.172, "Transmission Plan Aspects of International Conference Calls," Nov. 1988.

[73] P. T. Brady, "A statistical analysis of on-off patterns in 16 conversations," *Bell System Technical Journal*, vol. 47, pp. 73–91, Jan. 1968.

[74] J. Forgie, C. Feehrer, and P. Weene, "Voice Conferencing Technology Final Report," Tech. Rep. DDC AD-A074498, M.I.T. Lincoln Lab., Lexington, MA, Mar. 1979.

[75] J. D. Tardelli, P. D. Gatewood, E. W. Kreamer, and P. A. La Follette, "The benefits of multi-speaker conferencing and the design of conference bridge control algorithms," in *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 2, (Minneapolis,USA), pp. 435–438, Apr. 1993.

[76] P. Smith, P. Kabal, and R. Rabipour, "Speaker selection for tandem-free operation VoIP conference bridges," in *Proc. IEEE Workshop on Speech Coding*, (Tsukuba, Japan), pp. 120–122, Oct. 2002.

[77] M. A. Marouf and P. W. Vancil, "Method and apparatus for controlling signal level in a digital conference arrangement." United States Patent 4,499,578, Feb. 1985.

[78] M. A. Hashemi and G. P. Pucci, "Telephone conference system with active analog conference." United States Patent 4,139,731, Feb. 1979.

[79] N. K. Burns, P. K. Edholm, and F. F. Simard, "Apparatus and method for packet-based media communications." Canadian Patent Application 2,319,655, June 2001.

[80] K. Singh, G. Nair, and H. Schulzrinne, "Centralized conferencing using SIP," in *Proc. 2nd IP-Telephony Workshop (IPTel2001)*, (New York, NY), Apr. 2001.

[81] J. Rosenberg and H. Schulzrinne, "Models for multi-party conferencing in SIP." Internet Draft, Internet Engineering Task Force—Work in Progress, Nov. 2000.

[82] A. Shallwani, "An adaptive playout algorithm with delay spike detection for real-time VoIP," Master's thesis, McGill University, Montreal, Canada, Oct. 2003.

[83] D. Nahumi, "Delay synchronization in compressed audio streams." United States Patent 5,754,534, May 1998.