Integration of a Jitter Control Module into a Packet Voice Application

by

Wing Yee Winnie Lee
BASc Simon Fraser University, 2000

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING

in the School of Engineering Science

SIMON FRASER UNIVERSITY
November, 2003

Approval

Name: Wing Yee Winnie Lee

Degree: Master of Engineering

Title of Project: Integration of a Jitter Control Module into a Packet Voice Application

Supervisory Committee:

Dr. Jacques Vaisey
Senior Supervisor
Associate Professor
School of Engineering Science, SFU

Dr. James K. Cavers
Supervisor
Professor
School of Engineering Science, SFU

Date Approved: 2003 11 28

PARTIAL COPYRIGHT LICENCE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Project: INTEGRATION OF A JITTER CONTROL MODULE INTO A PACKET VOICE APPLICATION

Author: _____
(Signature)

_____
(Date Signed)

Abstract

In a packet voice application, source speech is coded and packetized and is sent through the network to the receiver, where the packet is decoded. The transfer of speech packets through a packet network introduces a variable transport delay. Packets may be lost and they may take different paths resulting in packets that arrive out of order. This arrival time variance (or jitter), packet loss, reordering and duplicating, must be handled properly to avoid gaps of degradation in the re-constructed speech. A jitter control module is required to cope with the asynchronous arrival of packets from the network.

A jitter control module is a key component of the speech service of a packet voice application. The packet received from the network will be held within a storage location referred to as the jitter buffer. The jitter control module is responsible for managing the release time of the packets stored within the jitter buffer to the speech decoder, where the speech will be reconstructed. The jitter buffer also works closely with the packet loss concealment algorithm (PLC) and the comfort noise generator (CNG) to ensure that the reconstructed speech is of the highest quality, while minimizing the end-to-end delay.

This document reports my work on the project – Integration of a jitter control module into the LDX at *Broadcom Canada LTD.* The project involves the integration of a jitter control algorithm into a Voice over Internet Protocol (VoIP) application with low channel density developed by Broadcom Canada called Low Density Xchange (LDX). The algorithm was originally developed for another Broadcom packet voice gateway solution with high channel density. The jitter control module was found to be vital for the quality of the reconstructed speech from LDX when operating over a jittery network. Data memory requirements have been optimized to ensure that the resource requirements of the jitter control module are acceptable on the target platform.

Acknowledgements

Thank you to Dr. Wilfred LeBlanc and Mr. Philip Houghton who provided me with good advice along the whole project. Thanks also to Dr. Jacques Vaisey and Dr. James Cavers for sitting on my committee and providing feedback on the project. Last but not the least, thanks to my family for their continuous love and support throughout my studies.

Table of Contents

List of Tables

List of Figures

List of Acronyms

IP – Internet Protocol

VoIP – Voice over Internet Protocol

LDX – Low Density Xchange

PVE – Packet Voice Exchange

DSP – Digital Signal Processing

MCPS – Million Cycles Per Second

PPM – Parts Per Million

A/D – Analog to Digital

D/A – Digital to Analog

VAD – Voice Activity Detector

SID – Silence Insertion Descriptor

CND – Comfort Noise Generator

PLC – Packet Loss Concealment

ECAN – Echo Cancellation

CDIS – Call Discrimination

PC – Personal Computer

QA – Quality Assurance

PCM – Pulse Code Modulation

UDP –User Datagram Protocol

PESQ –Perceptual Evaluation of Speech Quality

SDRAM – Synchronous Dynamic Random Access Memory

DMA – Direct Memory Access

ATM – Asynchronous Transfer Mode

1 Introduction

The field of packet voice communications over Internet Protocol (IP) networks has been growing rapidly in the recent years. In a packet voice application, source speech is coded and packetzied. The packetized speech is then sent through the network to the receiver, where the received packets are decoded so that the person on the receiver side can hear it. The success of a digital packet voice application relies on the provision of high speech quality while minimizing the end-to-end delay.

The following diagram explains how the encoded voice packets get transmitted from one end to the other in a packet voice application.

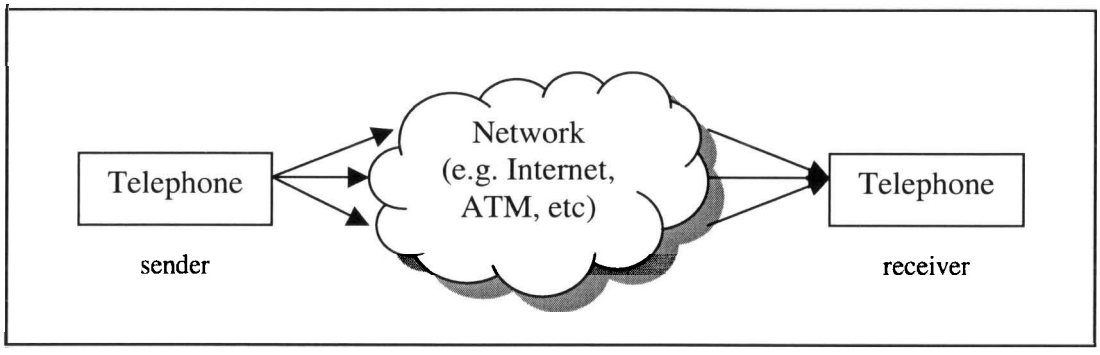

Figure 1: Transmission of encoded voice packets from one end to the other

The transfer of speech packets through a packet network introduces a variable transport delay. Packets may be lost and they may take different paths resulting in packets that arrive out of order. This arrival time variance (or jitter), packet loss, reordering and duplicating, must be handled properly to avoid gaps of degradation in the re-constructed speech. A jitter control module is required to cope with the asynchronous arrival of packets from the network.

1.1 Problem Statement

The purpose of this project is to integrate a jitter control algorithm into a Voice over Internet Protocol (VoIP) application with low channel density developed by Broadcom Canada called Low Density Xchange (LDX). The algorithm was originally developed

for another Broadcom packet voice gateway solution with high channel density. The target platform is Broadcom Canada's Hausware xChange Packet Voice Exchange (PVE) service, operating on the Broadcom BCM110x silicon family.

LDX is a suite of embedded Digital Signal Processing (DSP) algorithms for telecommunications applications. These algorithms can be combined to provide solutions of packet voice, fax and high-speed modem data over general networks.

1.2 Thesis Contribution

In a three-month period, I successfully integrated the jitter control module as part of the PVE service of LDX. Extensive testing has been done to prove that the newly integrated jitter control module provides a solution to cope with the network jitter in different situations. Minimal or no voice degradation is expected with the application of this jitter control module in LDX. Several optimizations have been done as an extension of the integration to minimize the data memory required by the jitter control module.

1.3 Thesis Organization

In Chapter 2, we go through basic theories of jitter and how it can affect voice quality in a packet voice application. It is followed by a description of how the jitter control module operates in the LDX Packet Voice Exchange service, and a high level description of the algorithm. Chapter 5 presents a detailed analysis of the performance of the jitter control module with some high level descriptions of the testing procedure. Optimizations to reduce memory requirement of the jitter control module is discussed in Chapter 6, followed by concluding remarks in Chapter 7.

2 Basic Theory of Jitter

Delay is a well-known problem that telephone network planners have had to manage since the early days of telephony [1]. Today's telephone networks have been designed to keep jitter effects imperceptible for most customers. However, when carrying voice over IP, it becomes much more difficult to control delay. Designing an acceptable service thus requires sophisticated technology and optimization of all components.

2.1 Media Path of IP Telephony

The media path of IP telephony calls can be modeled as shown in Figure 2

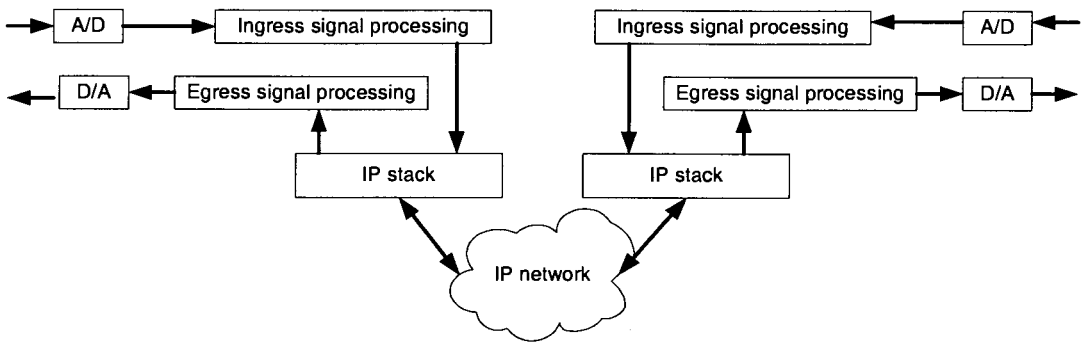

Figure 2: Media path of IP telephony calls

Figure 2 indicates a situation when a call is setup between two parties over a network. The LDX VoIP application works with digital speech samples provided by the analog to digital converter (A/D). The ingress signal processing of the VoIP application consists of different voice encoders. Incoming digital voice samples are compressed by one of the voice encoder algorithms into compressed voice frames. Compressed voice frames are sent to the network through the IP stack.

After encoded voice frames are transmitted across the network, they are received by the IP stack of the receiving party. The voice frames are then decoded by one of the voice decoder algorithms of the egress signal processing part of the VoIP application. The

reconstructed speech samples are converted back to analog signal by the digital to analog (D/A) device so the client can hear.

2.2 What is Jitter?

Jitter on a particular packet refers to the difference between the actual arrival time and the arrival time if the packet traversed the network with minimal delay.

$$Ta(n) = Ts(n) + Tm + J(n) \qquad \textbf{Equation 1}$$

where:

- Ta(n) = arrival time of the nth packet
- Ts(n) = send time of the nth packet
- Tm = minimum transit time over all n packets (i.e., minimum of Ta(n) – Ts(n) over all n)
- J(n) = random delay (jitter) associated with the nth packet

Jitter may change over time because of network traffic conditions. If jitter stays constant, we are referring to the situation when the random delay (jitter) associated with all the packets stays within a constant range throughout the call period. In other words, the difference between maximum and minimum delays is the same throughout the call period when the jitter stays constant. Normally, the maximum difference between the maximum and minimum delays is taken as the jitter encountered by the packets on the network.

In the absence of jitter, packets are received from a network connection at regular intervals, assuming Ts(n) is regular. The arrival of each packet at its destination is delayed from the time of its origination at the far-end of the network by the fixed network delay. It is reasonable to release each packet to the active decoder as soon as it is received so delay is minimized at the destination. This trivial case is illustrated below in Figure 3.

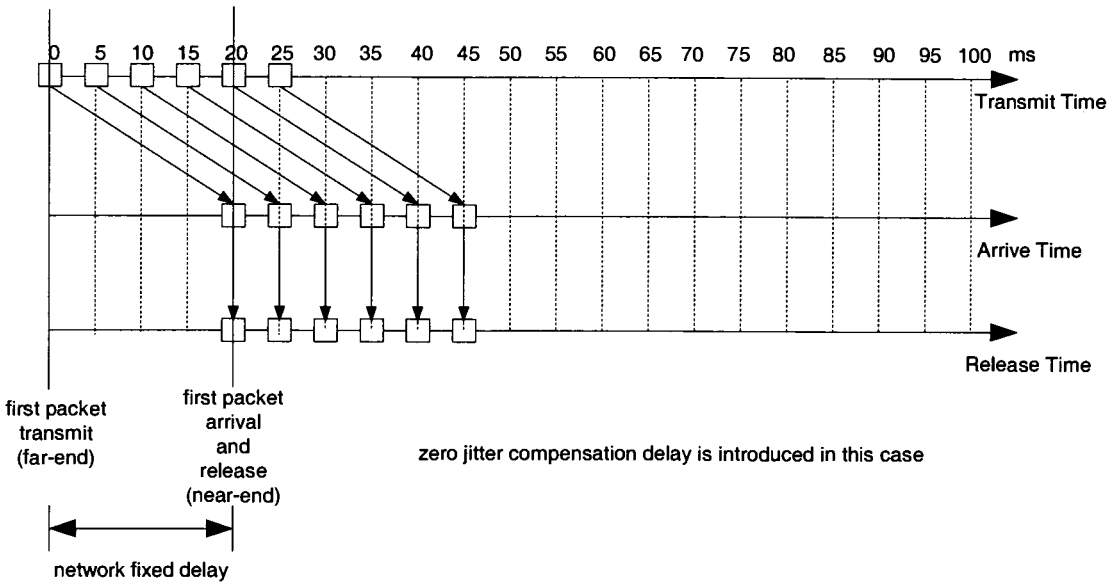

Figure 3: Packet timeline for zero jitter condition

The above condition will only happen in an ideal situation or on an isolated network. In normal situations, there is always some randomness in the propagation delay in addition to the fixed network delay, resulting jitter. The randomness in propagation may be a result of network traffic congestion, variation in processing time of different the IP stack, et cetera. In such a case, packets arrived with minimum delay must be held for a period at least equal to the known jitter.

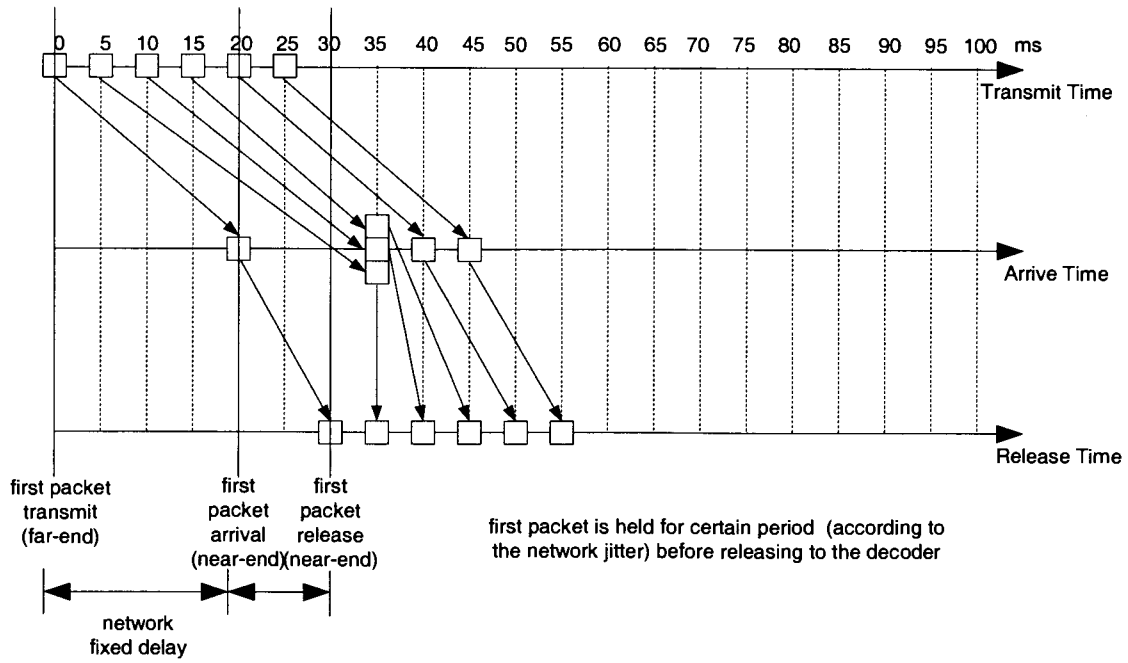

Figure 4: Packet timeline for known constant jitter

In Figure 4, the second packet sent to the network by the sender at time 5 ms has encountered an extra 10 ms delay compared to all the other packets. We say that there is 10 ms jitter associated with the network where the packets are sent across. If packets are released to the active decoder once they are received from the network, there will be a 10 ms gap (between time 25 to 35 ms) in the reconstructed speech.

In a packet voice application, packets are held in a place called the jitter buffer on the egress side. A jitter control module is used to control the release of the packets to the active decoder. If packets were held in the jitter buffer for duration shorter than the known jitter (10ms in the above example), an underrun would occur and there will be gaps in the reconstructed speech.

Sometimes packets will be sent through different paths to the destination, resulting in packets arriving out of order. The out of order packet appears to be "lost" but eventually arrives late. The situation of packets arriving out of order is illustrated in Figure 5.

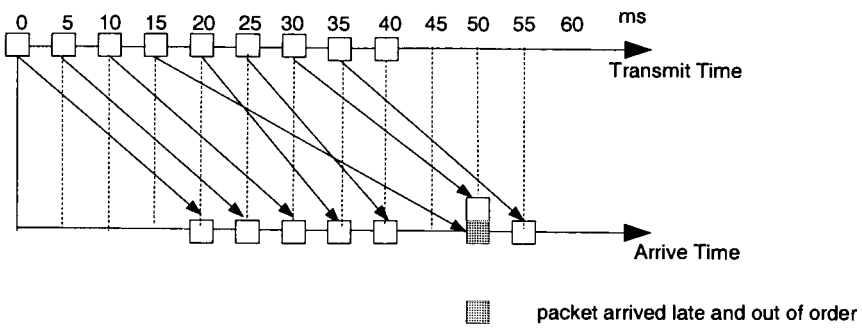

Figure 5: Packets arrive out of order

In Figure 5, the highlighted packet arrived late can either be deleted or inserted into the queue of packets stored in the jitter buffer, depending on the jitter buffer status. How the jitter control module handles out of order packets is dependent on the design specification of the particular voice application.

Other than the random delay caused by network transmission, clock drift is another type of source for variation of packet arrival time [3]. Clock drift is a result of system clock mismatch between the two systems involved in the call. The system with a faster clock will decode the packets at a faster rate compared to rate at which packets arrive from the network. The jitter buffer on the faster side is likely to underrun more frequently. On the other hand, the slower side will decode the packets at a slower rate compared to the rate at which packets arrive from the network, resulting in the building up of the jitter buffer.

Equation 1 is a simplified equation of the packet arrival time that ignores the effect of clock drift. If clock drift is take into account, the packet arrival time will change over time, regardless of the network condition.

$$Ta(n) = Ts(n) + Tm + J(n) + D(n) \qquad \textbf{Equation 2}$$

where $D(n)$ is the delay caused by clock drift. This delay changes with time and is determined by the amount of clock drift between the two calling parties. If the receiving

end is slower than the sender, `D(n)` will be smaller than `D(n+1)` because the sender is sending packets at a faster rate compare to the receiver's clock.

The variation in packet arrival time due to clock drift is less significant than the other components of jitter and it will only show up in calls with very long durations. The exact equation of determining the relation between `D(n)` and `D(n+1)` is complicated and is outside the scope of this document. The other components of jitter are discussed in the following sections.

2.3 Delay in a VoIP Telephone Network

Digital speech samples are compressed using different voice coders before being sent out across the network. Compressed speech is sent through the IP network in units called packets. The packets encounter five major types of delay [2].

1) packetization delay at the source
2) queuing delay at the IP stack
3) a fixed processing delay
4) transmission and propagation delay
5) a jitter compensation or depacketization delay at the destination

The following sections will provide brief descriptions of the different types of delay shown in the above list.

2.4 Packetization Delay at the Source

Most voice coders are frame oriented [1]. This means that they compress fixed-size chunks of linear samples, rather than apply compression on a per sample basis. Therefore the audio data stream needs to be accumulated until it reaches the chunk size, before being processed by the voice coder. This sample accumulation takes time, and therefore adds to the end-to-end delay. In addition some coders need to know more samples than those contained in the frame they will be coding (this is called lookahead).

In order to reduce delays on an ideal network, the chosen voice coder should thus have a short frame length. Unfortunately coders with larger frame sizes tend to be more efficient, and have better compression rates. Another factor is that each frame is not transmitted "as is" through the network – a lot of overhead is added by the transport protocols for each packet transmitted through the network. If each compressed voice frame is transmitted in a packet of its own, this overhead is added for each frame, and for some coders, the overhead will be comparable to if not greater than the useful data. To lower the overhead to an acceptable level, most VoIP applications choose to transmit multiple frames in each packet. A superpacket is a network packet with multiple voice frames concatenated in it. The idea of superpackets is illustrated in Figure 6.

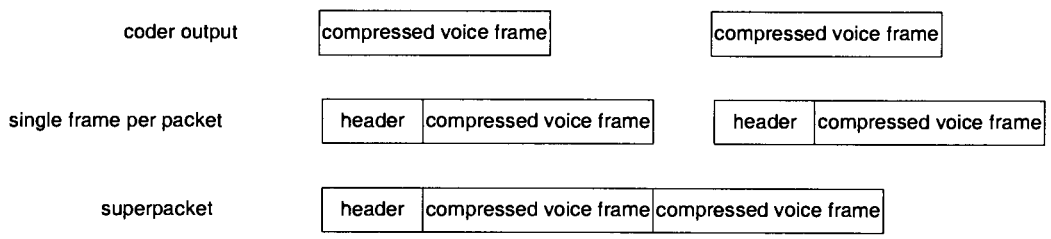

Figure 6: Superpacketization of encoded data

The packetization delay depends on the type of voice coder being used and the size of the superpacket being chosen. As long as the voice coder and the superpacket interval are unchanged, the packetization delay will remain constant.

2.5 Queuing Delay at Each IP Stack

A packet suffers queuing delay when there are other packets that arrived earlier or simultaneously to the IP stack and the Ethernet driver. It may need to wait in a queue until the stack and the driver finish processing the packets arrived earlier. The delay is random and depends on the traffic load and on the stack architecture [2].

2.6 Processing Delay

In addition to the queuing delay, the packet undergoes an almost deterministic processing delay at the sending and receiving end caused by the encoding and decoding process. Processing delay on the encoder side refers to the time needed by the processor to generate the voice frame provided that the raw speech samples are available. Processing delay on the decoder side refers to the time required by the processor to decode the received compressed voice frame from the network.

Although this processing delay is referred to as deterministic, it varies on processors with different speed and depends on the complexity of the active voice coder. For example, both G.728 and G.711 are 5 ms voice coders. G.728 encoder takes 28 million cycles per second (MCPS) while G.711 encoder only takes less than 0.35 MCPS. The processing delay of G.728 encoder is 1.4 ms while that of G.711 is less than 0.2 ms while running within a 5 ms thread on a 100 MHz processor.

The processing delay of the same encoder or decoder may have a slight variation, depending on the type of input signal. For example, the processing delay of the G.728 encoder is longer when the encoder is handling active speech signal, while the processing delay is shorter when the input signal is silence. The variation is caused by the difference in computation resource to handle different types of input signal; and less MCPS is required to handle silence in most cases. The variation of processing delay based on input signal only applies to complex voice coders such as G.729, G.728 and G.723.1, it is not applicable to stateless sample-based voice coders like G.711.

2.7 Transmission and Propagation Delay

Transmission and propagation delay refers to the time required to transmit the network superpackets from the source to the destination through the IP network. This delay is dependent on the network traffic as well as the quality of the equipment.

Among the above four types of delays described in Sections 2.4 to 2.7, only the queuing delay and the propagation delay are random and may change over time, these are the major sources of what we referred to as jitter in Section 2.2.

2.8 Jitter Compensation Delay

Other than network jitter and clock drift, some other network impairments include packet lost and packet duplication. In order to cope with all these network impairments, packets are held in the jitter buffer, resulting in delay for jitter compensation. How long the packets are held within the jitter buffer depends on the network conditions, the jitter control algorithm, as well as the size of the jitter buffer.

3 Structure of Packet Voice Exchange service

The Packet Voice Exchange (PVE) service is part of the LDX VoIP application. The service provides ingress and egress processing of the speech signal. Please refer to Figure 7 in the following sections.

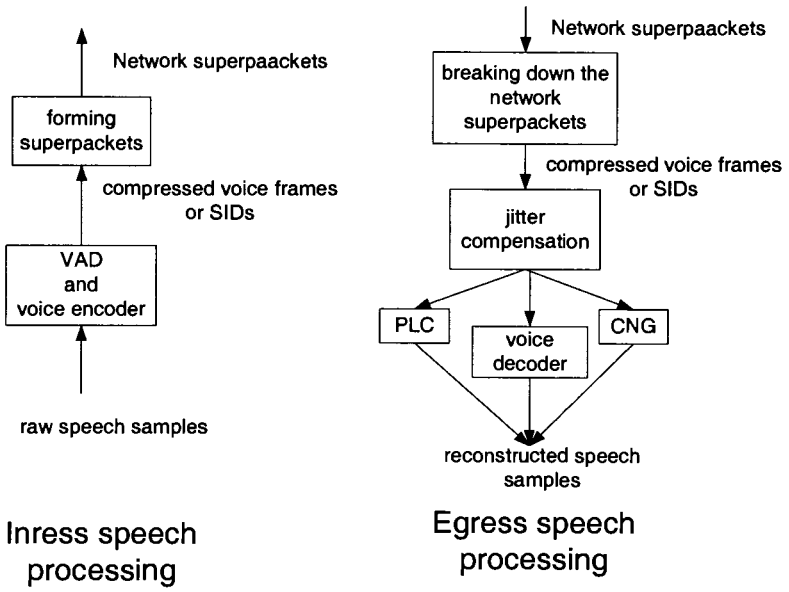

Figure 7: Ingress and egress signal processing of the PVE service

3.1 PVE Ingress Signal Processing

The digital raw speech samples are provided by the A/D converter. When enough raw speech samples have been accumulated, they will be sent to the PVE service for ingress signal processing. The Voice Activity Detector (VAD) will first examine the incoming voice signal. If the signal contains no active voice, a Silence Insertion Descriptor (SID) will be generated and the voice encoder will not be called. If the incoming signal contains active voice, the voice encoder will be called to generate compressed voice frame.

The individual compressed voice frames are sent to the packetization module of the PVE service, where they will be grouped together to form network superpackets (with

appropriate header information attached). The superpackets are now ready to be sent to the network.

The size of a SID is small compared to a compressed voice frame. Also, normal voice conversation is silent (contains no active speech) more than half of the time. As a result, using a VAD algorithm results in a more efficient use of the bandwidth available.

3.2 PVE Egress Signal Processing

When a network superpacket arrives at the PVE, it will first be broken down into native compressed voice frames by the depacketization module. The native frames will then be held in the jitter buffer for jitter compensation, where the jitter control module is responsible for controlling the release time of the voice frames.

The elements stored in the jitter buffer can either be SIDs or native compressed voice frames. There may be gaps in the packet stream as a result of packets lost or packet reordered. A Comfort Noise Generator (CNG) will be called when SIDs are received indicating that the far-end is having an inactive signal input. A voice decoder is called to decode the normal voice frames. In cases of other network impairments like lost or re-ordered packets, a Packet Loss Concealment (PLC) algorithm will be called to bridge the gaps in the decode speech samples.

Based on the elements stored in the jitter buffer and the algorithm being used, the jitter control module decides whether the CNG, the PLC or the voice decoder will be called. Linear Pulse Code Modulation (PCM) samples will be generated upon calling one of the three modules. Samples generated will be sent to the D/A converter for output.

4 Jitter Control Algorithm

As seen in Chapter 2, a jitter control module is required to handle the asynchronous arrival of packets from the network interface. The jitter control module must determine when to release speech frames to the speech decoder, when to play comfort noise, when to perform packet repeats to cope with lost frames or to extend the depth of the jitter queue, and when to perform packet deletions in order to decrease the depth of the jitter queue [4].

The challenge in jitter control is to ensure that the synthesized decoded voice is reproduced without delay variation whilst minimizing the end-end delay. These are competing priorities, and so a properly designed system along with an acceptable engineering trade-off between quality and delay is required.

Before we go over the high level description of the jitter control algorithm, there are some terms that are commonly used in discussions related to jitter.

4.1 Glossary Related to Description of Jitter Control Module

The far-end – is the originator of the network packets relative to the egress voice path where the jitter control module is situated.

The near-end – is the consumer of network packets relative to the egress voice path.

Packet timestamp – a number associated with each received packet that is generated by the far-end, which is incremented at regular intervals with respect to time and dependent on the active encoder being used.

Jitter buffer underrun – a condition where the jitter buffer is empty during an active speech period (non-silence), or when packets are expected.

Underrun duration – the duration of the period when the jitter buffer is empty until the arrival of the first packet following the underrun.

Jitter buffer overrun – a condition where either the number of packet exceeds the maximum number that can be stored in the jitter buffer, or the total playout time of the packets held in the jitter buffer exceeds the maximum allowable buffered playout time of the jitter buffer.

4.2 Jitter Control Module Operation Modes

A jitter control module usually has two modes of operation: variable holding time and fixed holding time.

4.2.1 Variable Holding Time

Adaptive jitter control is provided using holding time compensation and clock drift compensation. The holding time of packets in the jitter buffer changes with time, depending on the network conditions and clock drift between the calling parties. Packet deletions and packet repeats are used to adapt the packet holding times within the jitter buffer.

With an adaptive jitter control algorithm, jitter compensation delay discussed in Section 2.8 is longer when the packets encounter higher levels of jitter.

4.2.2 Fixed Holding Time

The fixed holding time mode provides no adaptive jitter control. Packets are held in the jitter buffer for fixed period of time, which is usually based on the worst-case jitter scenario. Packet deletions will only take place in an overrun condition. Packet repeats are used in an underrun condition. In this mode, the holding time of packets within the jitter buffer will not change regardless of the network conditions.

The fixed holding time mode of the jitter control module is used when quality of the decoded speech is the biggest concern. Since the packet holding time within the jitter buffer is based on the worst-case scenario, packet repeats or packet deletions are not likely to take place during the call because the jitter buffer is not likely to overrun or underrun. However, this fixed holding time mode results in long jitter compensation delay.

Under normal circumstances, an adaptive system offers a much better solution compared with a fixed system. This advantage is especially true in cases such as VoIP where the worst-case delay variation can be on the order of hundreds of milliseconds.

Delay measurements shown in Section 5.4 will indicate that an adaptive jitter control module is preferred in any network condition when small voice gaps during jitter buffer adaptation is allowed.

4.3 Performance Requirement of the Jitter Control Module

The primary goal is to maximize perceived voice quality under network changing conditions while minimizing either the end-to-end delay, or the jitter compensation delay. These are contrasting requirements, so a trade-off is required, especially since jitter is not something that is constant or time invariant.

Other important requirements are documented in Table 1:

Table 1: Requirements of the jitter control module

	Requirements
1	Rapid convergence of the depth of the jitter buffer queue.
2	Absolute minimum delay in stationary condition.
3	The jitter control module should be able to bride any silence gaps with no reconstructed phase jitter.
4	Bridging of gaps due to lost packets. This entails controlling the PLC and ensuring that there is no phase jitter.
5	Handling of out of order packets and redundant packets.
6	Robustness to packet timestamp anomalies, steps in delay and clock drift.
7	Under the condition of initial high level of network jitter followed by a period of low level of network jitter, the holding time of the packets within the jitter buffer should decrease at a rate of 1 ms per 1 second to minimize the jitter compensation delay. The decrease rate was chosen arbitrarily, based on the requirement that the jitter buffer size should not decrease too rapidly to minimize the glitches caused by packet deletes.
8	The holding time of the packets within the jitter buffer should increase quickly in response to increased network delays, which will result in underrun conditions. The holding time of the packets should increase with a step size as large as the underrun duration, limited by the depth of the jitter buffer.
9	Under the condition of maximum jitter, the jitter control module should aim for a holding time of packets within the jitter buffer less than or equal to the network jitter.
10	The adaptive jitter control module should have a mode of fixed packet holding time, as described in Section 4.2.2, for situations where voice quality is critical and no packet repeat or packet deletion is allowed.
11	The jitter control module should compensate for clock drift between 0 to 200 parts per million (ppm).

	Requirements
12	The jitter buffer must tolerate all packet arrival scenarios occurring due to the use of VAD on the far-end. Adaptive behavior should not be dependent on whether the far-end VAD is activated.

4.4 Design Issues of the Jitter Control Module

Major issues for consideration in the design of the adaptive jitter control module are discussed in Sections 4.4.1 to 4.4.3.

4.4.1 Release Time of First Packet

The decision of when to release the first packet held within the jitter buffer is paramount and is based on the estimated level of network jitter made using the timestamps of received packets. The packet with the earliest timestamp is released to the active decoder after being held in the jitter buffer for a predefined target holding time. This packet can be released earlier if subsequent arrivals can prove that the first packet was actually delayed.

4.4.2 Holding Time Adaptation

The holding time of the packets in the jitter buffer should adapt quickly to jitter buffer underruns, since this scenario indicates that the network delay is at a higher level than estimated by the jitter control module. On the other hand, the holding time of the packets within the jitter buffer should slowly reduce when the jitter control module estimates that the packets are being held too long in the jitter buffer compared to the current network jitter.

Packet repeats and packet deletions are used to increase or decrease the packet holding time within the jitter buffer respectively. Packet repeats will result in the PLC being called to bridge the gaps within the reconstructed speech. Whenever possible, packet deletions and repeats should be deferred until a period of relative silence to minimize the

impact on perceived voice quality. Extending the silence gaps from, for example, 1.2 seconds to 1.205 seconds is inaudible.

The speed of adaptation of the packet holding time within the jitter buffer is listed in the jitter control module specification in Section 4.3. However, the adaptation speed of the jitter control module in different applications may vary based on different customer requirements. For example, some customers prefer the holding time to increase slowly and to decrease quickly. This particular configuration avoids holding time increases when the jitter is occasionally bursty, with a tradeoff in voice quality when the jitter buffer underruns due to a jitter burst. The jitter control algorithm must be flexible enough to cope with different customer requirements.

4.4.3 Debug Information Provided by the Jitter Control Module

The jitter control module should provide debug statistics including packet repeats and deletions information. Jitter control module statistics are important to service modules such as the Echo Canceller (ECAN) and the Call Discriminator (CDIS). Jitter control module statistics will be discussed in detail in Section 4.10.

4.5 Key Variables within the Jitter Control Module

The key variables within the jitter control module include:

- `cT` – the current holding time of packets within the jitter buffer. This variable contains an up-to-date estimation of the worst-case jitter.
- `maxT` – maximum holding time of packets observed throughout the call duration.
- `minT` – minimum holding time of packets observed throughout the call duration.

The minT and maxT parameters are used as long-term indicators of jitter. Both variables are used to determine if cT needs to be adjusted. Upon start up condition, cT will be set to a predefined register value (set by the client) and adapts up or down from there. The predefined register value is set based on known network conditions. If the network conditions are unknown, cT can be initialized to zero, which means the first arrived packet will be released right away. The jitter control module should be able to increase or decrease cT based on the network jitter condition regardless of the predefined register setting. Having a correct predefined register setting eliminates the gaps in the synthesized speech caused by jitter buffer adaptation upon startup. Once the jitter control module is synchronized, the three parameters, cT, minT and maxT, should remain relatively constant as long as the network conditions are stable.

4.6 An overview of the Jitter Control Module algorithm

An overview of the jitter control module algorithm is provided in Sections 4.6.1 and 4.6.2.

4.6.1 Unknown network conditions

Assuming that the current network conditions are unknown, the packet holding time (cT) will be initialized to zero and the packet that arrives first will be released right away. If the network jitter is non-zero, then the jitter buffer will underrun when the packets are being held not long enough in the jitter buffer. cT will increase based on the underrun duration as described by the requirements number 8 stated in Table 1. After the jitter control module is synchronized, all parameters should be stable unless the network conditions change, and cT and maxT should be very close to (if not the same as) the network jitter level. Tracking parameter minT should be very close (if not equal) to zero, indicating the packets that experienced the longest network delay are just held long enough in the jitter buffer to avoid underrun. If the network conditions change and the network jitter level decreases, then the tracking parameter minT will be non-zero and the packet holding time within the jitter buffer should decrease at the rate specified in

requirement number 7 of Table 1. Test results indicating how `cT`, `maxT` and `minT` work are provided in Section 5.3.1.

4.6.2 Known network conditions

If the network conditions are known, the client can preset a register value to be used as the initial packet holding time within the jitter buffer. Upon start up condition, `cT` will be set to the predefined register value and adapts up or down based on requirements stated in Table 1. `cT` will never decrease below the predefined register value because it is assumed that this is the known network conditions and the end-to-end delay caused by this particular jitter buffer size is accepted. However, it will increase when the jitter buffer underruns, which indicates that the preset register value is too low and jitter buffer size increase is necessary to maintain voice quality. In cases where cT has been increased, it is possible to decrease it based on the same algorithm as described in 4.6.1 when the network condition changes and the network jitter level decreases.

Presetting the initial packet holding time of the jitter buffer helps prevent gaps in the synthesized speech in the beginning of a call caused by the holding time adaptation. However, gaps cannot be avoided if the preset value is too low, meaning that jitter buffer underrun can still occur. In addition, the end-to-end delay will be undesirably high if the initial packet holding is preset to a value that is unnecessarily high. This means that the initial packet holding time should not be preset unless the user has a complete understanding of the network conditions. In most applications, it is recommended that the user should not preset the initial packet holding time and should allow the jitter control module to adapt depending on the network conditions. By default, the initial packet holding time is set to zero.

The algorithm of the adaptive jitter control module is summarized in Figure 8.

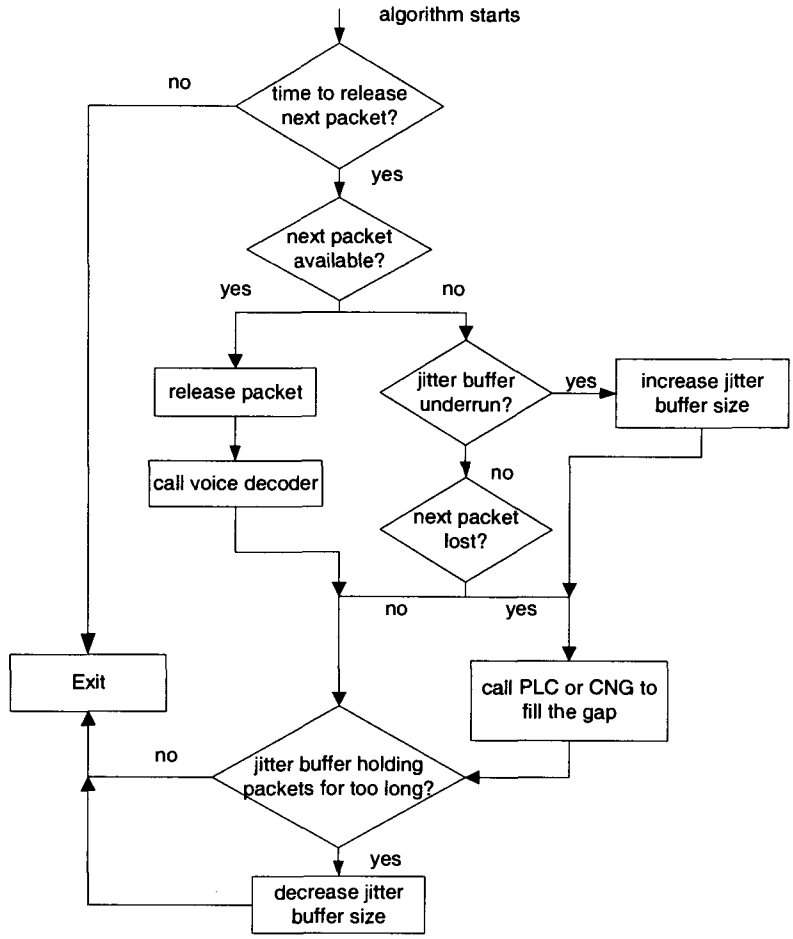

Figure 8: Block diagram of the adaptive jitter control module algorithm

4.7 Difference Between the Two Modes of Operation

As discussed in Section 4.2, the jitter control module has two modes of operation: variable holding time (adaptive) and fixed holding time. The adaptive mode is the operation mode when the jitter control module tracks the network jitter and adjusts the size of the jitter buffer accordingly. The fixed holding time mode is the operation mode when the holding time of the packets are fixed, packets will be held at least a certain amount a time in the jitter buffer before being played out by the voice decoder.

4.7.1 Jitter Buffer Underruns

The jitter buffer will underrun when voice packet is not available when expected. This means that the jitter buffer is not built large enough to handle the current network jitter.

When the jitter control module is operating in the adaptive mode, a jitter buffer underrun will occur whenever the network jitter increases to a level higher than the level estimated by the jitter control module. In other words, when the jitter control module is operating in the fixed holding time mode, the jitter buffer will only underrun when the network jitter is higher than the preset fixed packet holding time. Since the packet holding time of the fixed holding time mode is set based on the estimated worst case network jitter, a jitter buffer underrun will occur more likely in the adaptive mode. However, this is only limited to the beginning of a call when the jitter buffer is synchronizing to the current network condition. The jitter buffer should not underrun even when operating in adaptive mode after it has synchronized, unless the network condition changes.

4.7.2 Jitter Buffer Overruns

Jitter buffer overrun occurs when the memory allocated for the jitter buffer is all used up, or in order words, when the jitter buffer can no longer queue up the new incoming packets. This situation usually occurs when a burst of packets arrive from the network. With the current implementation of the jitter control module, each channel of the system is assigned a fixed sized array that forms the jitter buffer. The memory allocated for the jitter buffer is defined at compile time and will not be changed regardless of the run-time network condition.

If currently there is x ms or data stored within the jitter buffer, and the memory allocated can store y ms of data, an overrun situation will only occur when a burst of data bigger than $(y\text{-}x)$ ms arrives at the same time. This means that as the smaller the jitter buffer is built up, the larger the burst of packet it can receive at the same time.

Figure 9 summarized the memory usage of the jitter buffer managed by the jitter control module. The headroom mentioned in the figure is a measure of the size of the packet burst the jitter buffer can handle without overrun.

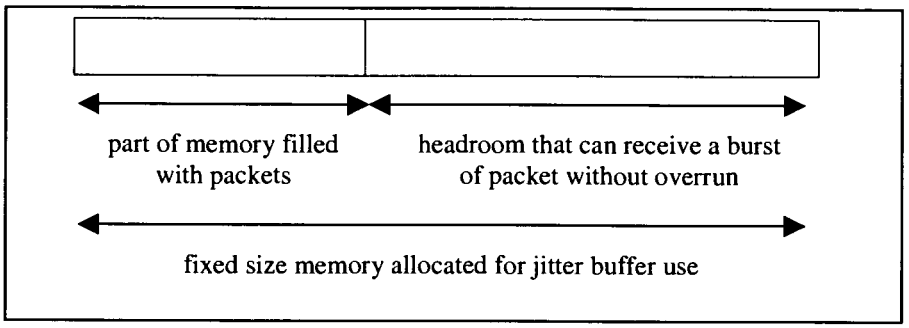

Figure 9: Memory Organization of the jitter buffer managed by the jitter control module.

When the jitter buffer is operating in the fixed holding time mode, the jitter buffer is expected to be built largest because the packet holding time is defined based on the known worst case jitter. In other words, there is a higher chance that the jitter buffer will overrun when the jitter buffer is operating in fixed holding time mode, when a big burst of packet arrive. However, the above only applies to the case when the actual network jitter is less than the worst case. When the jitter control module is operating in adaptive mode in network with the worst case jitter, the jitter control module will build up a large jitter buffer, jitter buffer overrun will still occur when a big burst of packets arrive at the same time.

4.8 The Size of the Jitter Buffer vs End-to-end Delay

The end-to-end delay of the voice path is dependent on components discussed in Sections 2.3 and 2.8, among which only the jitter compensation delay is affected by the jitter control module. The jitter compensation delay is dependent solely on the target holding time of the packets within the jitter buffer, as determined by the jitter control module based on the network conditions. If the jitter control module determines that the packets

should be held longer within the jitter buffer, a larger jitter buffer will be maintained and the jitter compensation delay will be longer as a result.

Basically, the size of memory allocated for the jitter buffer storage will not affect the jitter compensation delay, except that it limits the size of the jitter buffer, which only plays a part when the worst-case jitter exceeds the maximum allowed jitter buffer size. The memory allocated for the jitter buffer storage limits the maximum size of the jitter buffer, which in turn limits the maximum jitter compensation delay.

4.9 Details on Memory Allocated for the Jitter Buffer

When a jitter buffer is claimed to be an x ms jitter buffer, this means that when the jitter buffer has tracked the network jitter correctly, it can operate in a network with x ms of network jitter without overflowing or underrunning. When the size of the memory allocated for the jitter buffer storage cannot store the entire jitter buffer needed for the current network condition, overrun will happen and packets that cannot be stored will be discarded. The voice quality is expected to be poor in such a case because gaps will be introduced in the synthesized speech.

In order to support an x ms jitter buffer, the memory that is able to store two times x ms of data needs to be allocated. Two times the memory is required because we need to ensure that the jitter buffer will not overflow when the jitter buffer has adapted to x ms (storing x ms worth of data), while another burst of x ms of data arrive at the same time. The above situation would happen when suddenly the network condition changes and the all packets a transmitted through the network with minimum network delay (jitter level drops to zero ms).

4.10 Jitter Control Module Statistical Variables

Statistical variables are important in analyzing the status of the jitter control module. It also shows if the jitter control module is operating as expected. Table 2 summarizes the important jitter control module statistical variables.

Table 2: Jitter control module statistical variables

Variable Name	Description
peakHoldingTime	peak holding time since last statistics query
packetCount	the number of packets received from the network
addTailCount	the number of packets added to the tail of jitter buffer, this refers to the normal case when packets arrive in order
reorderCount	the number of packets arrived out of order
overrunCount	decoder overrun count, this refers to the time the jitter buffer is full
duplicateCount	the number of duplicate packets deleted.
outOfRangeCount	the number of packets with timestamps too far from the timestamps of the packets currently stored within the jitter buffer, this refers to the case when the jitter buffer is not storing enough packets for jitter compensation. The holding time (cT) is expected to increase in this case.
cantDecodeCount	the number of packets cannot be decoded, this refers to packets with a bad packet header
ajcUnderrunCount	the number of jitter buffer underruns
ajcDeleteCount	the number of packet deletes done to reduce packet holding time
ajcRepeatCount	number of packet repeats done to either increase holding time or due to lost frame
ajcResyncCount	the number of times the jitter buffer re-initialized. The jitter buffer usually re-initialized after overrun
ajcPhaseJitterCount	the number of times the jitter buffer inserted a phase discontinuity, this include doing a frame repeat or a frame deletes

4.11 Instance Memory Organization of PVE

The PVE service has to allocate data memory to save the state variables of the service. This chunk of memory is called the instance memory and is allocated on a per channel basis. The jitter buffer used by the jitter control module to store the incoming encoded voice frame is part of the instance memory of the PVE service. The organization of the PVE instance memory is summarized in Figure 10.

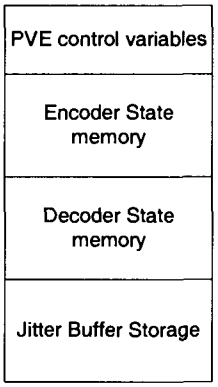

Figure 10: PVE service instance memory organization

The PVE control variables are variables and register values that control the operation of the service. This control variable set includes the configurations of the active encoder and decoder, the superpacket setting, the configuration of the jitter control module, et cetera. The size of this variable set is independent of the active voice coder.

The encoder state memory and the decoder state memory are instance memories of the active voice coder. The instance memory requirement is directly dependent on the complexity of the voice coder. For example, the encoder instance memory requirement of G.711 is only 2 bytes while the requirement of a G.729E encoder is 1986 bytes.

The memory required by the jitter buffer storage depends on the size of the jitter buffer supported by the jitter control module (the number of entries in the jitter buffer). The storage requirement is also dependent on the size of the encoded voice frames stored in the jitter buffer (the size of each entry in the jitter buffer).

5 Testing and Results

In Sections 5.1 and 5.2, we are going to discuss the two major testing environments we have implemented to test the newly integrated jitter control algorithm: simulation on a personal computer (PC) and real time testing in the Quality Assurance (QA) department.

Results from both environments have been utilized to verify the performance of the newly integrated jitter control module. However, due to limitations of the real time test setup in the QA department, we can only perform jitter test cases with periodic disruption using the real time test setup. The PC simulation environment will be used to perform more complicated testing: including jitter test cases with random disruptions and test cases with changing levels of network jitter.

Test results will be studied in Sections 5.3 and 5.4 based on three major aspects: statistical results based on the statistics variables of the jitter control module discussed in Table 2, voice quality measurement of the synthesized speech, and measured results based on the end-to-end delay of the voice path.

5.1 Testing Using PC Simulations

A PC application calling the C model of LDX was written. The application was written to provide a non-real time PC simulation environment for the developer. This software is particularly useful because running real time tests on the target platform is time consuming. The application is separated into two parts: ingress signal processing and egress signal processing.

The ingress signal processing part reads linear PCM data as input (to simulate the A/D converter). The speech service will then take this data and create encoded speech frames, which are written out to a file as output by the PC application. Timing information indicating when the encoded voice frames are generated is included in the output file. An example of an entry in the output file from the ingress signal processing simulation is shown in Figure 11.

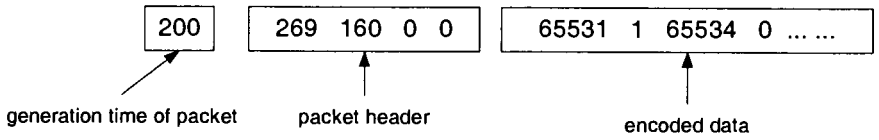

Figure 11: Output from ingress signal processing part of PC simulation

The egress signal processing simulation reads in compressed voice frames from a file as input. The format of the input file to the egress signal processing part is the same as shown in Figure 11. Timing information is included in the input file to indicate when is the time to read in a voice frame. The input voice frames will be handled by the speech service for jitter compensation and voice decoding. Synthesized speech will be written out to a file (in linear PCM format) for voice quality analysis.

In a jitter free scenario, the output file from the ingress processing simulation is used directly as the input file to the egress signal processing simulation. In this way, the packet generation time in the output file will be the time when the packet is being handled by the egress signal processing simulation.

The output file from the ingress simulation can be modified before being used as input to the egress simulation. The timing information in the first column can be modified (increased or decreased) to simulate network jitter. Also, entries in the file can be reordered, duplicated or deleted to simulate other network impairments. Individual test cases will be set up for different types of impairments to ensure that the jitter control module can handle different types of impairments. Test cases with all sorts of impairments applied at the same time will also be included to ensure that the jitter control module can handle different types of impairments at the same time.

Figure 12 shows an example of inserting network impairments to the output file from the ingress simulation. The left part of the figure is the original output file from the ingress processing simulation; and the right part of the figure represents the input to the egress simulation after network impairments have been inserted.

The first column indicates the timing information, represented in units of samples (at an 8 kHz sampling rate). The timing information has been modified to simulate a jittery network. The packet rate being used in the above example is 10 ms (80 samples at an 8 kHz sampling rate), so in a jitter-free test scenario, we are expecting packets to arrive at 10ms intervals, so the entries in the first column are incrementing at step sizes of 80 samples. The entries in the fifth column, highlighted in blue, are the timestamps of the encoded voice frames and are expected to increment at 80 samples intervals. As indicated in pink, if there is no jitter, encoded voice frame with packet timestamps 1360 and 2240 samples should arrive at time 1680 and 2520 respectively. The arrival time of the packets is delayed by 120 and 160 samples (15 ms or 20 ms) respectively after inserting 20ms jitter to the packet stream. Also an encoded voice frame with timestamp 1520 samples has been deleted from the file to simulate a lost packet scenario.

The second to the fourth column represents other information contained in the packet header, including the encoder being used, the packet type (whether it is an encoded voice packet or a SID) and the size of the packet. The data stated from the sixth column onwards indicates the encoded voice. The number of columns required to represent the encoded voice depends on the encoder being used.

Original output file from the ingress simulation

1400	271	320	0	1120	6	6	6	
1480	271	320	0	1200	65525	65526	65527	
1560	271	320	0	1280	65528	65527	65526	
1640	271	320	0	1360	6	6	6	
1720	271	320	0	1440	9	9	10	
1800	271	320	0	1520	65524	65525	65526	
1880	271	320	0	1600	65530	65529	65530	
1960	271	320	0	1680	7	7	6	
2040	271	320	0	1760	1	0	0	
2120	271	320	0	1840	65523	65524	65525	
2200	271	320	0	1920	1	1	0	
2280	271	320	0	2000	2	1	2	
2360	271	320	0	2080	65530	65531	65531	
2440	271	320	0	2160	65523	65523	65524	
2520	271	320	0	2240	3	3	4	

Input file to egress simulation with network impairments

1520	271	320	0	1120	6	6	6	
1520	271	320	0	1200	65525	65526	65527	
1600	271	320	0	1280	65528	65527	65526	
1800	271	320	0	**1360**	6	6	6	
1880	271	320	0	1440	9	9	10	
				1520				
2040	271	320	0	1600	65530	65529	65530	
2080	271	320	0	1680	7	7	6	
2120	271	320	0	1760	1	0	0	
2160	271	320	0	1840	65523	65524	65525	
2360	271	320	0	1920	1	1	0	
2360	271	320	0	2000	2	1	2	
2360	271	320	0	2080	65530	65531	65531	
2440	271	320	0	2160	65523	65523	65524	
2680	271	320	0	2240	3	3	4	

pink (1600, 1800, etc) - modified timing information
blue (1120, 1200, etc) - packet timestamps
red (1520) - timestamp of lost packet

Figure 12: Inserting network impairment to the output file from ingress simulation program

5.2 Real Time Testing as a Whole System

The test environment described in Section 5.1 is a non-real time simulation running on the PC. It is not sufficient to run tests on the PC only because the timing variation is different on the target platform. Also, many algorithms are coded in assembly language for efficient use of the DSP resources, while the PC simulation is running the C models of all the algorithms. This difference means that much of the code running in the real time system is not tested in the PC simulation. It is always necessary to run real time test on the target platform to ensure that the system is trouble free.

To test the newly integrated jitter control module, a test application that is able to generate network traffic congestion has been created. On the ingress signal processing part, different sizes of blank User Datagram Protocol (UDP) packets are generated by the test application. These blank UDP packets will be injected to the out-going packet stream generated by the LDX system to the network. The blank UDP packets are there to consume part of the available bandwidth and create network congestion. The blank UDP packets cannot simulate congestion caused by different network traffic, but other sources

of traffic congestion are not used in order to keep the testing environment simple. The blank UDP packets will be filtered out when they get to the egress side of the receiver running the same test application. Only the encoded voice frames will be passed down to the LDX system for egress signal processing.

A network emulation package that runs on Linux called NIST Net, provided by the National Institute of Standards and Technology, is being used in the above test environment [5]. The NIST Net network emulator is a general-purpose tool for emulating performance dynamics in IP networks. By operating at the IP level, NIST Net can emulate the critical end-to-end performance characteristics of the networks. The tool is designed to allow controlled, reproducible experiments with network performance sensitive or adaptive applications and control protocols in a simple laboratory setting.

In this test application, all packets, both encoded voice frames and the blank UDP packets, are sent across the network controlled by NIST Net. The NIST Net emulator is used to control the available bandwidth of the network. NIST Net uses a big internal buffer to hold all the packets it receives from the sender. The amount of data that it will send to the receiver is based on the configured bandwidth of the network under control. If the amount of data NIST Net received from the sender temporarily exceeds the available bandwidth, the extra data is stored temporarily within the NIST Net internal buffer, and those extra data will be sent to the receiver later when bandwidth is available. In this test application, blank UDP packets with different sizes can create different jitter scenarios based on proper NIST Net configurations. Figure 13 summarizes the test setup used to run the real time test on the jitter control module.

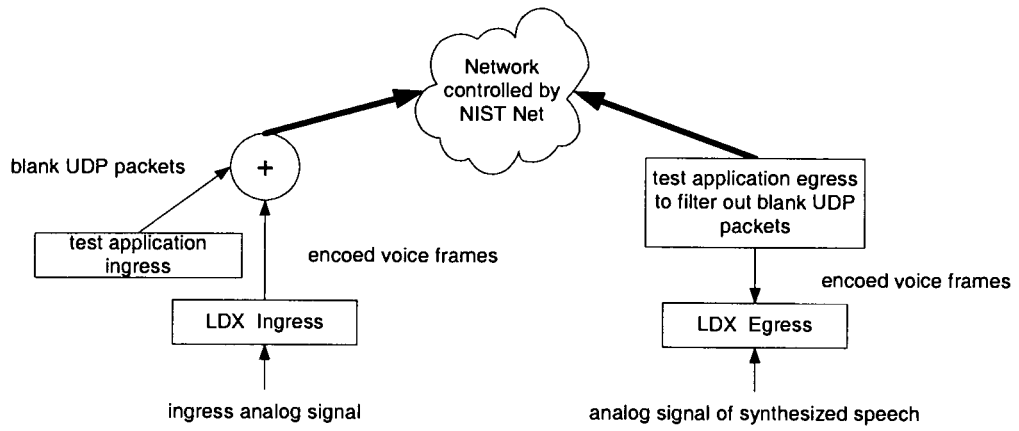

Figure 13: Architecture of real time test setup

Figure 14 shows the network traffic congestion caused by the blank UDP packets generated by the test application.

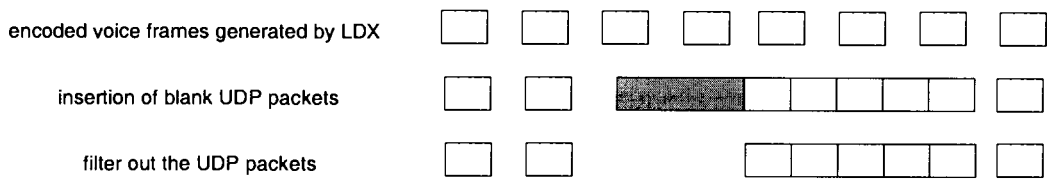

Figure 14: Effect of inserting blank UDP packet to cause network congestion

When a competing UDP packet is traveling across the network, it causes the encoded voice frames to be held up. Once the UDP packet has gone across the network, the buildup of the encoded voice frames are then flushed down the network. Therefore the amount of jitter is dependent upon the size and time that it takes for the blank UDP packets to travel to the destination.

The size of the blank UDP packets is configured by the desired amount of jitter based on the available bandwidth. Just as an example, if NIST Net provides 128 kbps of bandwidth, 1840 bytes of blank UDP packets will be generated at a 1 second interval to create approximately 120 ms of jitter when the superpacket interval is set to 20ms. Blank UDP packets of 1840 bytes are used because about 140 ms is required to transmit 1840

bytes of data at the 128 kbps bandwidth available. The packets are expected at 20ms interval in a jitter free scenario, the packets sent immediately after the blank UDP packet will experience 140ms delay, meaning that it is delayed by 120 ms extra. Packets following this particular packet will be blocked in the temporary buffer of NIST Net. After the blank UDP packet is sent to the destination, all the packets held in the NIST Net temporary buffer will be flushed down the network, meaning that some packets will experience a lower network delay.

Both the peak jitter and the distribution of jitter times are important: the peak jitter is important to ensure that the jitter control module provides an accurate estimation of the current network situation; the distribution of jitter times (when the blank UDP packets are sent) are important to ensure that the adaptation rate (especially the decrease rate) of the packet hold time is as expected. In a jitter free situation, packets should arrive at a regular interval, depending on the configured superpacket interval. With the test setup configured to generate approximately 120ms jitter and the superpacket interval set to be 20ms, the distribution of the inter-packet arrival time is shown in Figure 15.

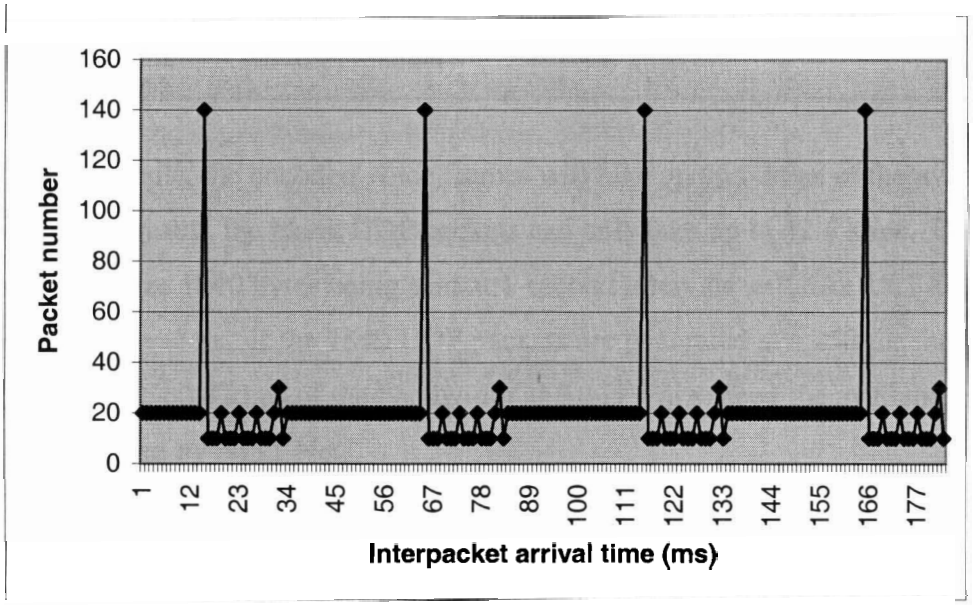

Figure 15: Inter-packet arrival time when the test setup is configured for 115ms network jitter

The above test case is used to simulate network situations when packets arrival is bursty, which is very common in an IP network. Other network situations are hard to simulate and those test cases are still under implementation.

In the test setup, NIST Net is configured to provide a network with a bandwidth or 128 kbps. We need to make sure that the bandwidth required by the encoded voice frames plus the blank UDP packets does not exceed the available bandwidth provided by NIST Net, or else too much jitter will be created. The bandwidth requirements of the encoded voice frames are listed as in Table 3. Note that G.711 20ms superpackets are used in this example.

Table 3: Bandwidth requirement of encoded voice frames

Network packet component	Bandwidth requirement
G.711 payload	64 kbps
RTP packet header (6 words per superpacket)	4.8 kbps
UDP packet header (4 words per superpacket)	3.2 kbps
IP header (10 words per superpacket)	8 kbps
Ethernet header (7 words per superpacket)	5.6 kbps

In the above example, the encoded voice frames will take up 85.6 kbps of the available bandwidth. As a result, the blank UDP packets can only take up to 42.4 kbps. Blank UDP packets of size 1840 bytes being sent at 1-second intervals will take 15.1 kbps including all the headers. If the 1840 UDP packets are generated at a 250 ms intervals instead, it will take 60.4 kbps of the bandwidth and this will exceed the available bandwidth provided by NIST Net.

Note that due to limitation in NIST Net and the test environment, the real time test setup for the jitter control module can only do simple test cases with periodic disruption as indicated in Figure 15, more complicated testing will be performed using the PC simulation test environment.

5.3 Test Results Analysis

Both C simulation and the real time test described in Sections 5.1 and 5.2 have been utilized to verify the performance of the newly integrated jitter control module. Three important measurements, the jitter control module statistics, the voice quality measurement and the end-to-end delay measurement are being used to analyze the test results.

5.3.1 Jitter Control Module Statistics

The jitter control module statistics described in Section 4.10 are important measures of the jitter control module performance. Based on requirements listed in Table 1, if the adaptive jitter control algorithm is being used, the packet holding time within the jitter buffer should quickly adapt and should be stable after it adapts, as long as the network jitter condition remains stable, which means that the jitter statistics are constant, and the network delay experienced by the voice packets only fluctuates within a constant range. Also, the packet holding time within the jitter buffer should never exceed the estimated network jitter. Packet deletes and packet repeats should be performed only during the holding time adaptation, after the jitter control module adapts, no packet repeats or deletes should be performed unless there is a packet lost scenario, or when the network condition changes.

Figure 16 is a plot of the peak packet holding time (*peakHoldingTime* in Table 2) and the tracker variable `minT` described in Section 4.5 within the jitter buffer while the test conditions are controlled by NIST Net as described in Figure 15, so the network jitter is controlled to be 120ms. *peakHoldingTime* and `minT` in Figure 16 are plotted at 1-second intervals when packets are arriving from a network with 120 ms jitter. The network conditions have not been changed throughout the call, so it is expected that *peakHoldingTime* and `minT` should remain at about 120 ms after it adapts. The adaptation parameters used by the jitter buffer is listed in Table 1. Based on the specification listed in Table 1, *peakHoldingTime* should not decrease throughout the call

because the blank UDP packets are sent at constant interval throughout the call to create jitter.

Except in the first second when the jitter control module is adapting to the network jitter, *peakHoldingTime* stays at 110 ms throughout the call. The packet holding time adapts up by performing packet repeats. In this test, the G.711 voice coder with a 5 ms frame rate, was used and 22 packet repeats were recorded to adapt the packet holding time from 0 ms to 110 ms. After the jitter buffer size adapts to 110ms, no further frame repeats or frame deletes was observed. Also, note that the tracker variable `minT` stays 0 throughout the call duration, meaning that the packets are held just long enough in the jitter buffer and the jitter control module is tracking the network jitter accurately.

Note that *peakHoldingTime* does not read 120ms but only read 110ms instead is because of effect of superpacket interval is taken into account by the jitter control module. With voice data received in units of 20ms superpacket, 15 ms jitter is created because the native frame rate of G.711 is only 5ms and the packets now come in burst of 20ms superpackets. The jitter control module considers the effect of superpackets (15ms) by reducing the *peakHoldingTime* reading accordingly by 15ms, leaving the *peakHoldingTime* reading to be 110ms (round up from 105ms) instead.

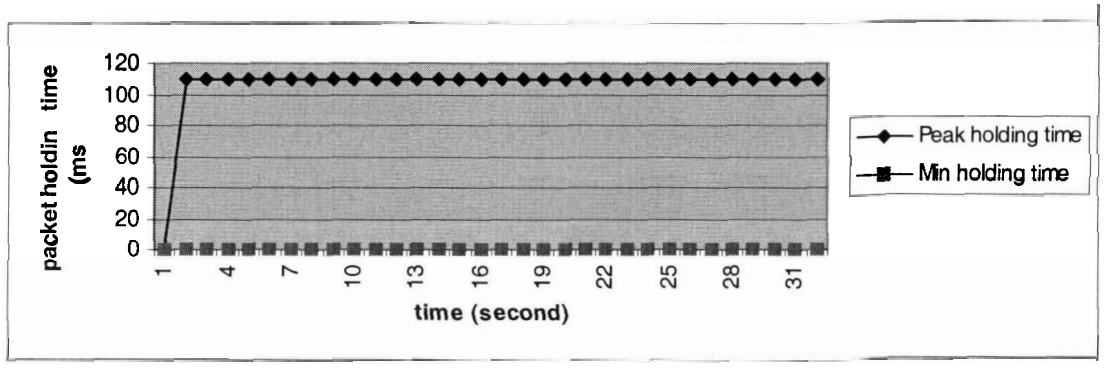

Figure 16: Packet holding time within jitter buffer at 115ms jitter

The jitter control module statistics from the PC simulation matches the statistics recordings from the real time test when the above same test was run.

The packet holding times vary depending on different network jitter conditions. For example, the holding time should adapt to and remain stable at a higher level in a test case with higher level of network jitter, or the packet holding time should decrease if the network jitter decreases in the middle of the call.

Figure 17 is a plot of the peak packet holding time (*peakHoldingTime* in Table 2) and the tracker variables `minT` and `maxT` described in Section 4.5 within the jitter buffer while the test condition are controlled by the PC simulation as described in Section 5.1. The network jitter is aimed to be changing throughout the call. Within the call under test, 20 ms jitter was inserted to the call for 1 second, after that the jitter level was decreased to zero. The jitter level stays zero for 20 seconds and them it was bumped up to 50 ms. The jitter level only stayed at 50 ms for another second before it was decreased back to zero again.

As we can see from Figure 17, the peak holding time tracks the simulated network jitter accurately. The adaptation rate of the packet holding time and the packet deletion mechanism for jitter buffer adaptation (when the jitter level decreases) was discussed in Section 4.6.1.

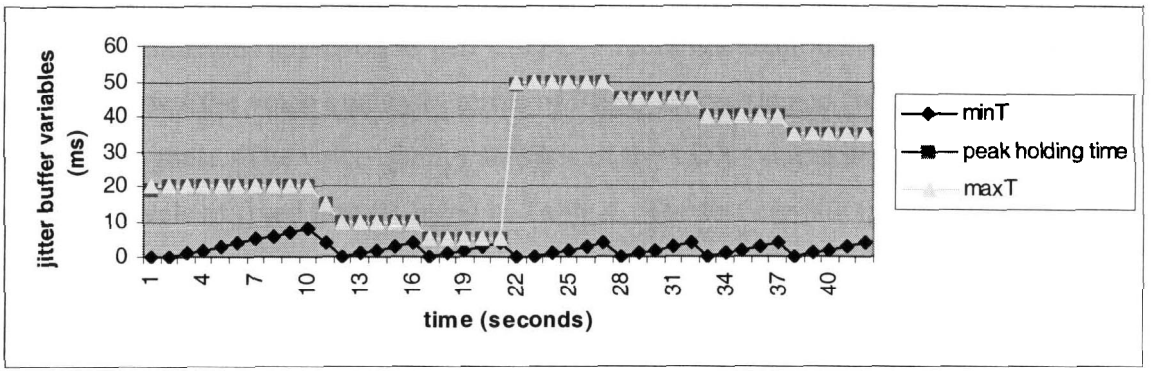

Figure 17: Jitter control variables when network jitter is changing

The jitter control module statistics are used in other test cases to ensure that the specifications on jitter buffer adaptation rates and the expected performance in other

network impairments are satisfied. The test case described above is only part of the test performed on the jitter control module. Other test cases to stress the system is still under construction and verification.

5.3.2 Voice Quality Measurement

The jitter control module statistics only verifies that the requirements of the jitter control module listed in Table 1 are satisfied. However, problems sometimes exist in the modules interacting with the jitter control module. Voice quality is a critical measurement in determining the success of the voice application and is included as part of the jitter control module testing. As before, voice quality analysis can be performed on the PC simulation as well as on the real time system.

In the real time test, a node plays a reference speech stimulus file to the A/D of the LDX system. The speech signal is encoded and the encoded voice frames are sent across the bandwidth-limited network with the blank UDP packets present to simulate congestion. The LDX system on the receiving side performs jitter compensation and decoding on the incoming packets. Synthesized speech is recorded and voice quality of the degraded speech is validated using the Perceptual Evaluation of Speech Quality (PESQ) method [6] [7]. PESQ is an enhanced perceptual quality measurement for voice quality in telecommunications approved as part of ITU-T recommendation P.862. The PESQ analysis shows the voice quality in terms of PESQ scores ranged from 0 to 4.5, from lowest to highest. The voice quality analysis of the LDX system in a jittery network with different levels of fixed jitter is listed in Table 4. The test conditions are controlled by the blank UDP packets and NIST Net in the same way as described in Figure 15. The jitter control module was set in adaptive mode and adaptation specifications listed in Table 1 are used to generate the following results.

Table 4: Voice quality analysis of real time jitter test

voice coder	jitter free	25 ms	50 ms	115ms	135ms	175ms
G.711-ulaw	4.3	4.24	4.24	4.24	4.24	4.24
G.711-alaw	3.99	3.97	3.97	3.97	3.94	3.97
G.726 16 kbps	3.17	3.16	3.16	3.16	3.17	3.17
G.726 24 kbps	3.85	3.84	3.83	3.84	3.82	3.83
G.726 32 kbps	4.19	4.14	4.15	4.15	4.15	4.12
G.726 40kbps	4.28	4.22	4.23	4.23	4.23	4.23
G.729A	3.83	3.81	3.83	3.83	3.83	3.82
G.729E	4.06	4.04	4.04	4.04	4.03	4.04
G.728	3.92	3.91	3.90	3.91	3.90	3.90
G.723.1 6.3 kbps	3.74	3.76	3.75	3.75	3.76	3.75

Table 4 summarizes the result of voice quality measurement when the jitter control module is configured to operate at adaptive mode as described in Section 4.2.1; the voice quality measurement is exactly the same when the jitter control module is configured to operate at fixed mode as described in Section 4.2.2. This means that the voice quality is not affect by the different modes of operation the jitter control module.
As we can see from Table 4, the voice quality is not affected by the fixed level of jitter on the network. Only the end-to-end delay has been increased because the jitter compensation delay has been increased as the jitter control module adapts to higher jitter levels. Please refer to Section 5.4 for details on end-to-end delay measurements.

The above test case only stresses that the jitter control module is able to track different levels network jitter accurately. Other test cases to stress the jitter control module in drastic network condition is still under construction. It is expected that the voice quality will decrease due to gaps in the synthesized speech when the jitter level is high enough to cause jitter buffer to overflow/underflow.

To show the effect of the jitter control module, non-real time PESQ analysis has been performed on synthesized speech of the PC simulation when the jitter control module is

disabled. By disabling the jitter control module, this means that the decoder will play out the latest incoming voice frame received from the network. Timestamp information of the packets will not be considered. If more than one encoded voice frame arrived from the network at the same time, only the voice frame arrived last within the frame tick will be decoded, earlier arrivals will be discarded.

The results of the analysis are shown in Table 5. For simplicity, only G.711 u-law voice coder is used in this analysis.

Table 5: Voice quality analysis when jitter control module is disabled

jitter level	PESQ
0 ms	4.4
25 ms	1.47
50 ms	0.81
115 ms	0.2
135 ms	0.12
175 ms	0.12

As we can see from Table 5, the jitter control module is vital in providing speech with acceptable quality in a VoIP application.

The configuration of the real-time test case is quite limited and the same configurations needed to be used throughout the voice call, this means the network conditions need to remain stable throughout the voice call in the real-time test cases.

5.4 End-to-end Delay Measurement

Other than the jitter control module statistics and the voice quality measurement, the end-to-end delay of the voice path is also an important measure of the jitter control module. Since the real time network conditions are difficult to control, the PC simulation environment is used for measuring the end-to-end delay of the voice path.

The network jitter (and also the simulated jitter inserted in the simulation environment) only results in fluctuations of packet arrival times, the average packet arrival time, or the average end-to-end delay of the signal path is unchanged by the jitter level, provided that the jitter compensation delay mentioned in Section 2.8 is not in the picture. As a result, the end-to end delay of the voice path is directly related to the holding time of the packets within the jitter buffer. If the minimum end-to-end delay of the simulation environment in a jitter free scenario is x milliseconds, the end-to-end delay of the voice path when y milliseconds of network delay is inserted should be equal to x plus y milliseconds. If the measured end-to-end delay of the voice path is less than x plus y milliseconds, then the packets are not being held long enough within the jitter buffer and underruns are expected. On the other hand, if the end-to-end delay of the voice path is more than x plus y milliseconds, this means that packets are held unnecessarily long within the jitter buffer, which is also undesirable.

The PESQ voice analysis tool provides an option for delay measurement. Delay measurements can also be performed on a per utterance basis so the variations in the end-to-end delay throughout the call can be analyzed.

5.4.1 End-to-end delay when jitter level is stable

Figure 18 shows the end-to-end delay on a per utterance basis of a speech file with 42 utterances when the jitter level was zero. Each utterance is approximately 1 second in length and the same speech file is used in the following measurements for easy comparison.

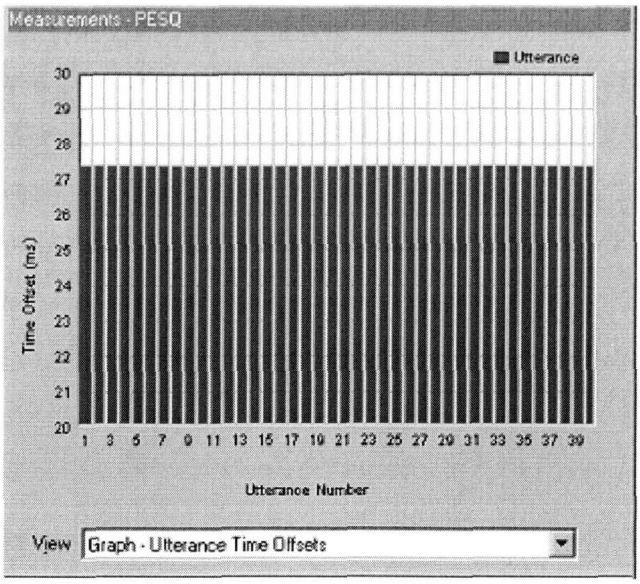

Figure 18: End-to-end delay when jitter level is zero

Figure 18 shows that the minimum end-to-end delay of the PC simulation model is about 27 ms and this quantity will be used as the guideline to show the depth of the jitter buffer in the following measurements.

Figure 19 and Figure 20 are end-to-end delay measurements when the jitter level is set to 30ms and 120ms. As indicated in Figure 19 and Figure 20, the end-to-end delay of the voice path is 57ms and 147 ms when the network jitter is 30ms and 120ms respectively. This means that the packets are held just long enough in the jitter buffer since the end-to-end delay is 27ms in a jitter free scenario.

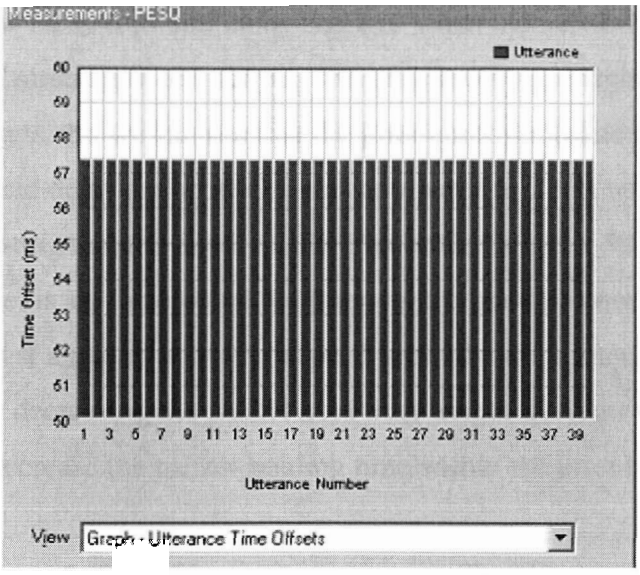

Figure 19: End-to-end delay when network jitter is at 30ms

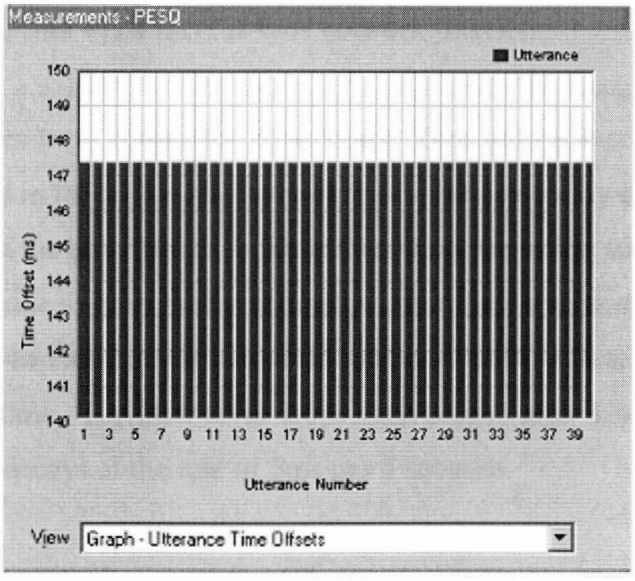

Figure 20: End-to-end delay when jitter level is at 120ms

5.4.2 End-to-end delay when network conditions are changing

In this particular test case, the PC simulation test script is implemented to insert different levels of network jitter at different time and requirements listed in Table 1 are used in this test case.

Figure 21 shows how the end-to-end delay changes when the network jitter is changing as controlled by the PC simulation test script. The jitter level was originally set to 30ms when the call just starts and we can see that the jitter control module tracks that correctly because the end-to-end delay is about 60ms (as compared to 27ms when there is 0ms network jitter, indicating that the depth of the jitter buffer is about 30ms). After 1 second, the jitter level is set to zero and the jitter control module controls the end-to-end delay to drop slowly at a rate of about 5 ms per 5 seconds (each utterance is about 1 second in length) by decreasing the packet hold time within the jitter buffer. Packet deletes are used to decrease the packet holding time within the jitter buffer.

The jitter level stayed at 0ms for 20seconds, this means that the jitter control module only has time to decrease the jitter buffer size by 20ms. As we can see in Figure 21, the end-to-end delay only drops to about 37 ms, meaning that the jitter buffer size is still about 10 ms (as compared to 27ms when there is 0ms network jitter).

After leaving the jitter level at 0ms for 20 seconds, 50ms jitter is inserted into the packet stream. As indicated in the figure, the jitter control module quickly picks it up by increasing the size of the jitter buffer to 50ms after the jitter buffer tracks the increased network jitter level (due to jitter buffer underflow after the network delay has been increased). The end-to-end delay is increased to about 80ms, indicating the depth of the jitter buffer size is 50ms. The jitter level drops back to zero after 1 second, and the end-to-end delay slowly decays at the rate of 5ms per 5 seconds.

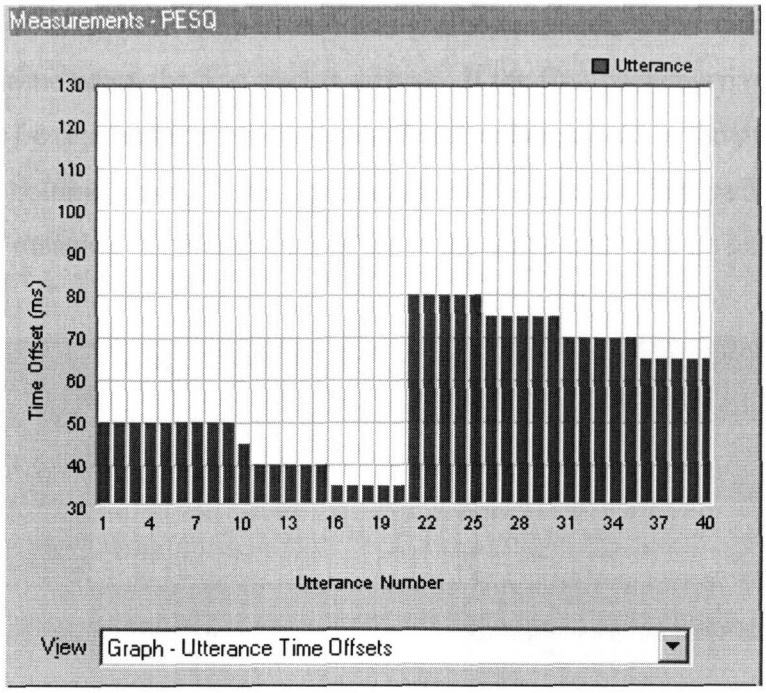

Figure 21: End-to-end delay when jitter level is changing

5.4.3 Adaptive versus fixed jitter control module

As discussed in Section 4.2, the jitter control module has two modes of operation: variable holding time (adaptive) and fixed holding time. The adaptive jitter control module is preferred in most cases because it correctly tracks the network jitter and packets will not be held unnecessarily long in the jitter buffer as compared to the fixed mode jitter control module configured for the worst-case jitter scenario.

Figure 22 and 21 show the end-to-end delay of the voice path when the jitter control module is in the fixed mode configured at 150ms target size. We can see that the end-to-end delay is longer than that seen in Figure 19 and Figure 20 when the jitter control module is in adaptive mode.

Also, as seen from Figure 22 and Figure 23, we can see the end-to-end delay is not directly related to the network jitter. The end-to-end delay of the voice call when the jitter level is 120ms is only about 30 ms longer than the case when the jitter level is

50ms. This result is because the packet holding time within the fixed mode jitter buffer depends on the time when the first packet arrives. If the first packet arrives early, the first packet will be released to the decoder early (after being held in the jitter buffer for the fixed target holding time). If the first packet arrives late, the first packet will be played out late, meaning the end-to-end delay of the voice path will be increased.

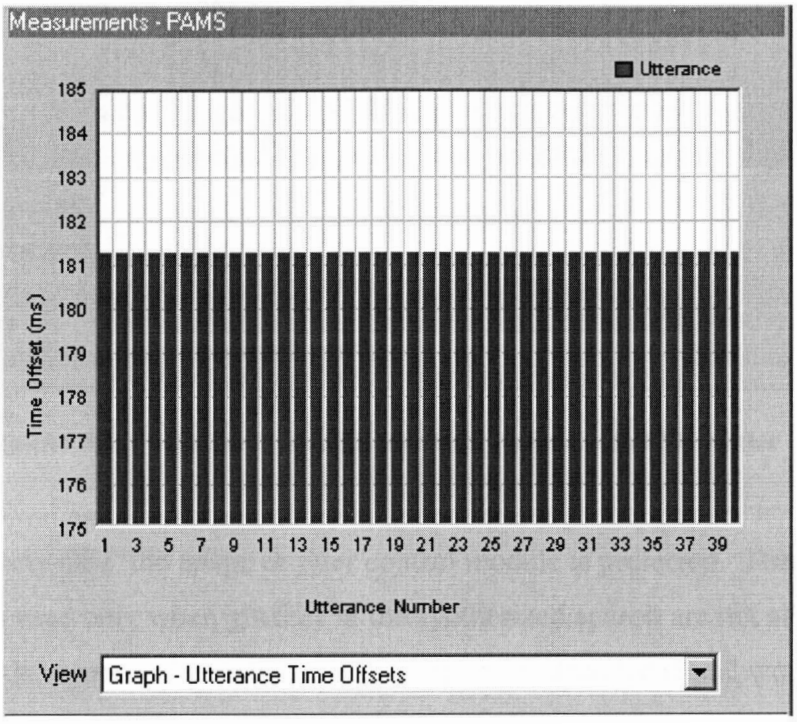

Figure 22: End-to-end delay with fixed mode jitter control module with 50ms jitter

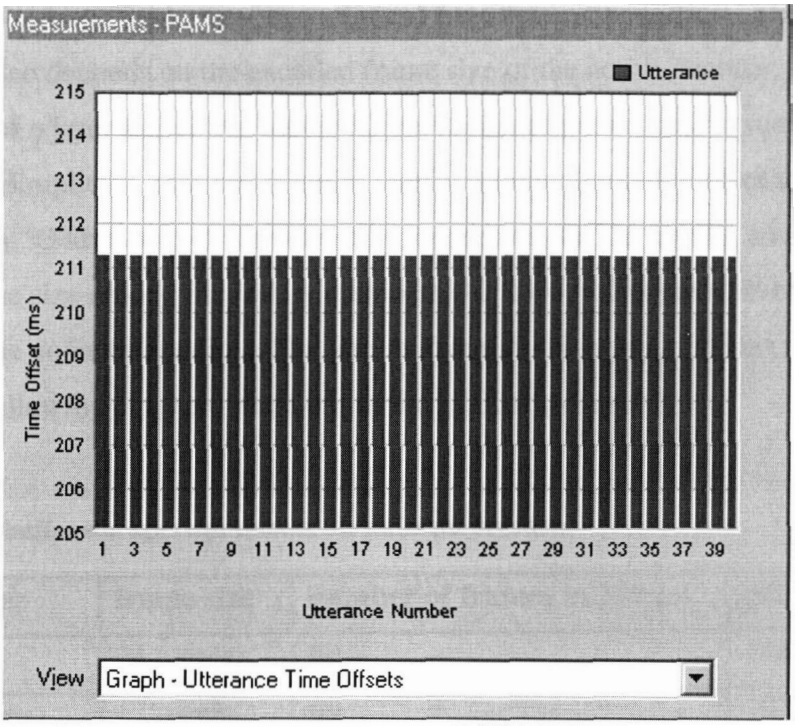

Figure 23: End-to-end delay with fixed mode jitter control module with 120ms jitter

As a result, in any case, the adaptive jitter control module is preferred. The fixed mode jitter control is used only when glitches in the synthesized speech are not allowed, as in the case of modem connection, when a single gap in the decoded signal would result in a modem disconnect when the modem is not configured to support error correction.

5.5 Jitter Control Module Resource Requirements

5.5.1 Memory Requirements

The program memory requirement of the jitter control module is about 3000 words of program memory. The data memory required by the algorithm itself is minimal because it only consists of several lookup tables.

The data memory required for the jitter buffer storage is allocated on a per channel basis, with the size of the jitter buffer depending on the level of network jitter supported by the application. The jitter level on an IP network is typically in the order of hundreds of

milliseconds; so large jitter storage is required for jitter compensation. The size of the jitter buffer also depends on the encoded frame size of the active decoder. The memory requirement of a jitter buffer that stores 300ms worth of packet data is summarized in Table 6. For simplicity, only the decoder family is being quoted. For example, G.726 16 kbps, 24 kbps, 32 kbps and 40 kbps are all grouped into G.726 family, and the largest encoded frame size among the family members (40 kbps in this particular example) will be taken as the voice frame size. The jitter buffer size for storing 300 ms worth of data is used in the following example since this is the LDX requirement.

Table 6: Jitter buffer storage requirement for different decoders

Voice decoder	frame size	number of frames in 300 ms	jitter buffer size
G.711	24 words	60	1440
G.726 (40 kbps)	17 words	60	1020
G.729 (E)	12 words	30	360
G.728	9 words	60	540
G.723.1 (6.3 kbps)	16 words	10	160

With the PVE instance memory organization shown in Figure 10, the largest requirement will be allocated for each component of the PVE instance memory. As a result, 1440 words will be allocated for jitter buffer storage in each channel since G.711 is the default voice coder for LDX.

5.6 Computation Resource Requirements

The computation cycle requirement of the jitter control module is not big and varies depends on the network jitter level. More computation cycles will be required when the jitter control module is maintaining a large jitter buffer while the requirement is minimal when the jitter buffer is of zero size. On average, the jitter control module is using less than 1.5 MCPS. This number is acceptable, since the LDX is a low channel density voice application on a target platform with more than 100 MCPS of computation resources. An average ITU standard complex voice decoder will take an average of about 5 MCPS on

the target platform. Although the MCPS consumption of the jitter control module is significant compared to that of the speech coder, optimization is not of top priority because the target platform has available resources to handle the load.

6 Optimization

Optimization in the instance memory usage of the PVE service is a very important portion of this project. Optimization has been done in two major areas so that the jitter control module can run efficiently on two different types of platforms supported by LDX. Two types of target platforms are supported by LDX: the platform on which the DSP has access to external Synchronous Dynamic Random Access Memory (SDRAM) and the platform on which the DSP has no access to external memory.

6.1 Optimization on Platforms with Access to External Memory

6.1.1 Implementation on the BCM1100 Platform

The BCM1100 chip is one of the target platforms supported by the LDX. The BCM110x silicon family is a single-chip system with integrated MIPS and ZSP DSP processors, which is designed specifically for low-density VoIP applications. Both processors on the BCM1100 have access to a common SDRAM. The ZSP processor has direct access to a small internal memory, while it has access to the slow external SDRAM at the same time. However, access time to the external memory is very slow compared to ZSP internal memory. A Direct Memory Access (DMA) is available to ease the transmission of data in and out of the SDRAM on the ZSP. The MIPS processor has direct access to the SDRAM, so no DMA is necessary for modules running on the MIPS.

The PVE service is separated into two parts on the BCM1100 architecture: the MIPS part and the ZSP part. The part of the PVE service running on the MIPS handles all the network packets before passing them to the part of the PVE service running on the ZSP. The part of the PVE service running on the ZSP is responsible for jitter compensation and decoding the voice frames for synthesized speech.

The internal memory map of the ZSP processor is only 16K words and, as indicated in Table 6, the large storage requirement of the jitter buffer is the limiting factor for channel density. The LDX running on the BCM1100 architecture stores the whole jitter buffer

for each channel on the SDRAM. Each time the jitter control module runs, the whole jitter buffer is brought in to the ZSP internal memory for jitter compensation. After the jitter control module has done its job, the whole jitter buffer is brought back to the SDRAM until the next time the control module runs again.

Although DMA is being used in the memory transmission, bringing such a large piece of data memory in and out of the SDRAM still requires many computation cycles and this becomes the limiting factor for channel density. On average, about 3 cycles are required to transmit 1 word of memory from the external SDRAM to the fast internal DSP memory on the target platform; the same amount of computation resources are required for the reverse direction. As a result, 8640 cycles are required to transmit the jitter buffer from and back to the SDRAM each time the jitter control module is called. The jitter control module is called at the same rate as the current active decoder. For example, if G.711 is being used, the jitter control module and the G.711 decoder are called at a 5 ms intervals, which means that 1.728 MCPS will be required for the memory transmission to and from the SDRAM for each channel. This large overhead is not acceptable since the jitter control module algorithm itself is taking less than 1.5 MCPS.

An entry in the jitter buffer is a native encoded voice packet, consisting of a packet header and the encoded speech. The jitter control module only studies the packet headers for controlling the release time of the packets and jitter adaptation. The encoded speech is not needed until the packet gets to the decoder for playout. As a result, the jitter buffer storage can be split into two sections, the header section and the encoded speech section, with each section residing in different areas of memory. The header section is stored in the fast internal memory for jitter compensation; the encoded speech is stored in the slow external SDRAM. The jitter buffer storage will be re-defined as stated in Figure 24.

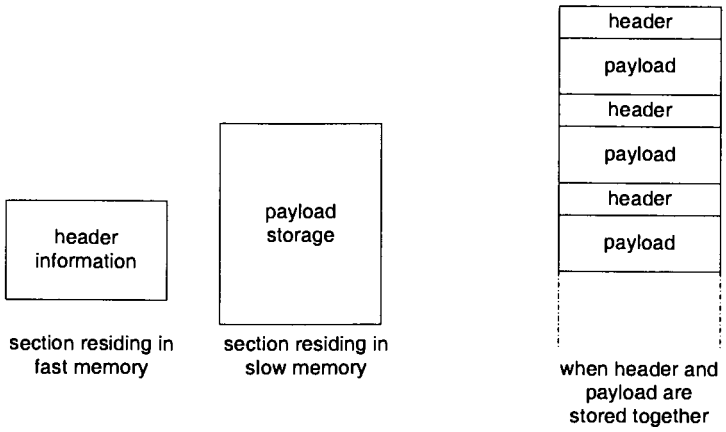

Figure 24: Jitter buffer storage split into fast and slow sections

When the header and payload are stored together, the whole jitter buffer needs to reside in fast memory so the jitter control module can refer to the packet header information. The memory required for the header information is small (4 words for G.711 packet headers) in comparison to the memory required for the whole packet (24 words for G.711 encoded speech with header).

When the PVE running on the MIPS receives a packet from the network, it first breaks up the network superpacket into the native voice packets. The encoded speech part of each native voice packets will be written to the shared SDRAM, only the header part is sent down to the ZSP PVE jitter control module. The ZSP PVE performs jitter compensation based on the header information and accesses the SDRAM for the encoded speech of that particular packet when the packet needs to be decoded. The egress voice signal-processing path on the BCM1100 chip is summarized in Figure 25.

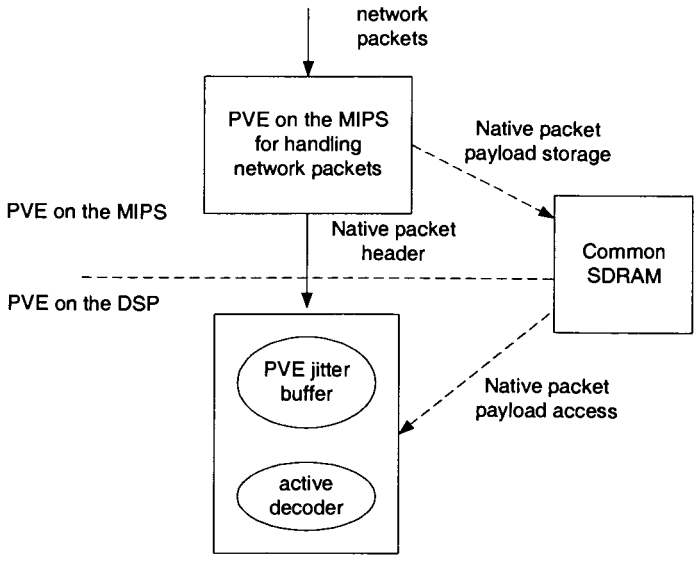

Figure 25: Egress voice signal-processing path on BCM1100

With this implementation, only the header information is stored in the ZSP internal memory and the memory requirement of the jitter buffer is now 240 words, a reduction of 1200 words.

To reduce the requirement of the ZSP internal memory, this jitter buffer containing the header information could also be stored in the external SDRAM. However, the amount of data memory brought in and out of the SDRAM reduces from 1440 words to 240 words each time the jitter control module runs, and the computation cycles spent on DMA is greatly reduced from 1.728 MCPS to 0.288 MCPS.

6.1.2 Extension to other Platforms with Access to External Memory

The above implement can be extended on any single processor chip with access to external memory. The PVE service will not be split in this case and the network superpackets will get directly to the ZSP. After the superpacket is split into the native encoded speech frames by the PVE service, the encoded speech data will be written to the external memory while the header information will be retained in internal memory for jitter compensation. This implementation can be summarized in Figure 26.

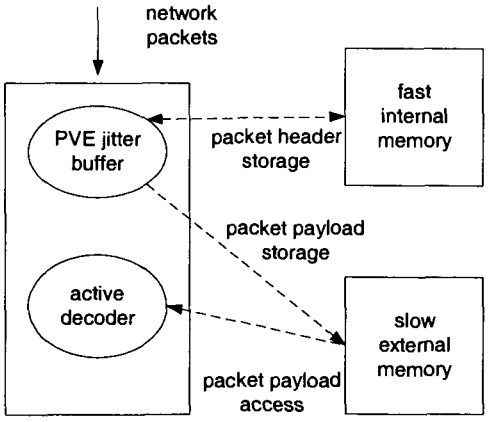

Figure 26: Egress voice signal-processing path on single processor chips with external memory

6.2 Optimization on Platforms without Access to External Memory

When the jitter control module is running on a platform with no access to external data memory, the header and the encoded speech data will be stored together in a single jitter buffer as indicated in the right most model of Figure 24.

As shown in Figure 10, the instance memory required by the egress signal-processing part of the PVE service is composed of two main components:

- the decoder state memory
- the jitter buffer storage

Both the size of the jitter buffer and the decoder state memory are directly related to the complexity of the decoder. When the decoder is more complex, the decoder state memory will be bigger to store the more complex decoder state variables. However, the jitter buffer storage requirements for complex decoders are usually smaller because complex voice coders provide better voice compression. When the PVE instance memory is organized as shown in Figure 10, the largest decoder state memory required by the most complex decoder and the largest jitter buffer required by the least complex decoder will be allocated as part of the PVE instance memory. The instance memory

organization for the PVE egress signal processing service can be summarized in Figure 27.

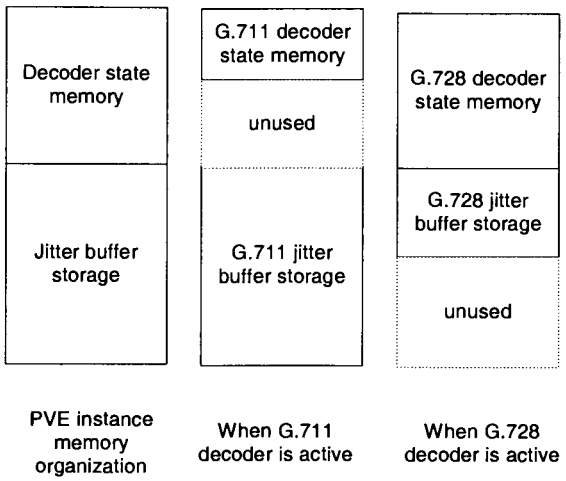

Figure 27: Original organization of PVE egress signal processing instance memory

Table 7 describes the sizes of the jitter buffer and the corresponding state memory for all decoders LDX currently supports. Again, the jitter buffer storage requirement is based on a jitter buffer that stores 300 ms worth of encoded speech.

Table 7: PVE egress signal-processing instance memory requirement

Voice decoder	jitter buffer	decoder state memory	combined requirement
G.711	**1440**	42	1482
G.726 family	1020	98	1118
G.729 family	360	832	1192
G.728	540	**1038**	**1578**
G.723.1 family	160	208	368

As we can see in Table 7, the instance memory requirement of the PVE service egress signal-processing part in the original implementation is 2478 words (1440 words of G.711 jitter buffer plus 1038 words of G.728 decoder state memory). In the new

implementation, a sliding boundary is implemented between the decoder state memory and the jitter buffer storage. Each time a new decoder is being used, the boundary between the decoder state memory and the jitter buffer will be reset for that particular decoder. With the new implementation, only 1578 words is required for each channel of PVE egress service, a net saving of 900 words per channel is achieved. Also, the amount of unused instance memory at any time can be significantly reduced. The above implementation can be summarized in Figure 28.

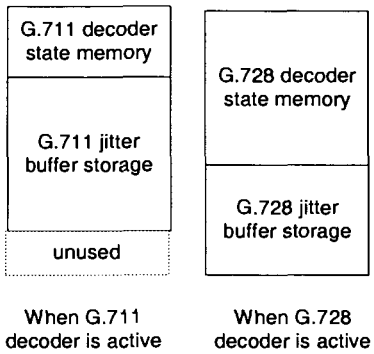

Figure 28: New organization of PVE egress signal processing instance memory

With this new approach, the jitter buffer will be flushed and the PVE egress instance memory will be re-initialized every time the PVE receives a new encoded voice frame that does not belong to the current active decoder. The boundary between the jitter buffer and the decoder state memory will be reset properly for the new active decoder.

The jitter buffer will be flushed every time there is a decoder rate switch. All the existing packets within the jitter buffer will be deleted. A glitch will be resulted in the synthesized speech during decoder rate switch. However, this problem is not critical because there will be phase differences when the decoder changes.

7 Conclusions

The jitter control module is a powerful tool to effectively improve quality of synthesized speech while maintaining minimum end-to-end delay. Jitter control modules have been widely adopted in voice applications targeted for different types of network including IP, Asynchronous Transfer Mode (ATM), et cetera. The purpose of this project was to integrate the jitter control module that was originally implemented for Broadcom gateway application with high channel density into LDX, a Broadcom voice application with low channel density.

In this document, I explained the necessity of the jitter control module for voice applications running on normal network environment with fluctuations in packet arrival times. Details of the theory on network jitter and a brief introduction of the jitter control module algorithm are provided in Chapter 3 and 4.

Testing is an important portion of this project and a lot of effort has been done to verify the performance of the new jitter control module. Both jitter control module statistics and voice quality analysis are utilized to validate the algorithm's performance. The test results ensure that the jitter control module is vital to the quality of the synthesized speech of the VoIP application.

A critical portion of this project is to optimize the data memory requirement of the jitter buffer. With details discussed in Chapter 6, we figured out that re-organizing the data memory storage of the jitter control module, including splitting the jitter buffer memory into fast and slow memory portions, or merging the jitter buffer with the decoder state memory, all help to reduce the data memory requirement on a per channel basis.

Optimizing the program memory requirement of the jitter control module should be an exciting next step. The jitter control module was originally implemented on a high channel-density gateway platform where program memory requirement is not a limiting factor. There is a lot of repetitive code that was intentionally written to reduce the MCPS

requirement of the jitter control module. Optimizing these repetitive routines should reduce the program memory requirement at the cost of increasing the MCPS requirement.

References

[1] Oliver Hersent, David Gurle and Jean-Pierre Petit, "IP Telephony – Packet based multimedia communications systems". *Pearson Education Limiter,* 2000, Chapter 4.

[2] Jean Walrand and Parvin Varaiya, "High-performance Communication Networks", *Morgan Kaufmann Publishers Inc.*, 1996, pp 203-207.

[3] Altmann Micro Machines, "Welcome to Jitter.de", www.jitter.de/english/engc_navfr.html, accessed: April 30th, 2003.

[4] Broadcom Canada Ltd. LDX document, "Adaptive Jitter Control Module Design", 2002.

[5] National Institute of Standards and Technology, "Nist Net Home Page", http://dns.antd.nist.gov/itg/nistnet/, accessed: April 17th, 2003.

[6] British Telecommunications, "News about PESQ", http://www.pesq.org/, accessed April 30th, 2003.

[7] International Telecommunication Union Standardization Sector, "Recommendation P.862 – Perceptual Evaluation of Speech Quality (PESQ), an objective method of end-to-end speech quality assessment of narrow-band telephone networks and speech codecs", February 2001.