HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Laboratory of Acoustics and Audio Signal Processing

Martti M. Mela

Utilizing DSP for IP Telephony Applications in Mobile Terminals

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, April 28, 2006

Supervisor: Professor Paavo Alku
Instructor: Jari Selin MSc

For my father, Professor Martti Mela (1933-2005)

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Martti M. Mela
Työn nimi:	Signaaliprosessorin hyödyntäminen mobiilin laitteen Internet-puheluohjelmistoissa
Päivämäärä:	28. huhtikuuta 2006 **Sivuja:** 77
Osasto:	Sähkö- ja tietoliikennetekniikka
Professuuri:	S-89 Akustiikan ja äänenkäsittelytekniikan laboratorio
Työn valvoja:	Professori Paavo Alku
Työn ohjaaja:	DI Jari Selin

Tässä diplomityössä etsitään ja määritellään optimaalinen ohjelmistoarkkitehtuuri reaaliaikaisen puheenkoodauksen mahdollistamiseksi mobiilin laitteen Internet-puheluohjelmistossa. Arkkitehtuurille asetettiin vaatimus, jonka mukaan puhelu ja siihen liittyvä puheen reaaliaikaisuus ei saa rajoittaa tai liikaa kuormittaa laitteen muuta toiminnallisuutta.

Työssä käytetty mobiili laite tarjoaa mahdollisuuden hyödyntää kahta prosessoria. Toinen prosessoreista on tarkoitettu yleisille käyttöjärjestelmille sekä ohjelmistoille ja toinen signaalinkäsittelyoperaatioille. Suunniteltu arkkitehtuuri yhdistää näiden kahden prosessorin toiminnallisuuden ja mahdollistaa reaaliaikaisen puheenkoodauksen (sekä toisto että äänitys) mobiiliisissa laitteessa.

Arkkitehtuuri toteutettiin ja sen suorituskykyä arvioitiin erilaisilla mittauksilla ja parametreilla. Havaittiin, että toteutus suoriutuu erinomaisesti sille asetetuista vaatimuksista. Todettiin myös, että käytettäessä ainoastaan laitteen yhtä prosessoria reaaliaikavaatimus ei täyty. Tämä johtuu puhekoodekin matemaattisesta kompleksisuudesta ja laitteen rajoitetuista ominaisuuksista.

Työn aikana jätettiin kaksi patenttihakemusta.

Avainsanat: AMR, DSP, Internet-puhelu, OMAP, reaaliaikainen puheenkoodaus, signaaliprosessori, RTP, RTSP, SIP, Sofia-SIP

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Martti M. Mela
Name of the Thesis:	Utilizing DSP for IP Telephony Applications in Mobile Terminals
Date:	April 28, 2006 **Number of pages:** 77
Department:	Electrical and Communications Engineering
Professorship:	S-89 Laboratory of Acoustics and Audio Signal Processing
Supervisor:	Professor Paavo Alku
Instructor:	Jari Selin MSc

In this thesis, an optimal software architecture is studied and defined for enabling a real-time speech coding scheme in an Internet telephony application of a mobile terminal. According to a requirement set for the architecture, a phone call and the related real-time speech coding shall not limit or overload other functionality of the terminal.

The mobile terminal utilized in this thesis provides a potential to take advantage of the efficiency of a dual core processor. One of the processors is designed for general purpose operating systems, and the other one for signal processing operations. The designed software architecture combines the functionality of these processors and enables real-time speech coding (both playback and capture) in the device.

The architecture was implemented and its performance was evaluated with different measurements and parameters. It was observed that the implementation outperforms the requirements set. It was also confirmed that the performance of the general purpose processor is inadequate for real-time operations with the chosen speech coder/decoder.

Two patent applications were filed by the author during the writing of this thesis.

Keywords: AMR, DSP, Internet telephony, OMAP, signal processor, real-time speech coding, RTP, RTSP, SIP, Sofia-SIP

Acknowledgements

I would like to thank Professor Paavo Alku for his guidance and comments for writing and finalizing this thesis. I am also grateful for Professor Matti Karjalainen for his inspirational teaching and support he gave to me while writing this thesis.

At Nokia Research Center I have enjoyed the bright atmosphere and dark coffee for years. My gratitude goes to my instructor Jari Selin for sharing his wide knowledge and going deeply into the thesis. Aki Niemi has been a great manager and his support was essential in completion of the studies. Toshihiro Kobayashi's help was invaluable for the utilization of the DSP Gateway software. I want to thank Kai Vehmanen and Pekka Pessi who have willingly shared their time and knowledge. It has been an exciting opportunity to build the Sofia-SIP library together.

I also want to thank my friends. Without you I would have graduated already several years ago.

Finally, I want to thank my girlfriend Stéphanie for introducing me to Confiserie Sprüngli.

Otaniemi, April 28, 2006

Martti M. Mela

Contents

Abbreviations

3G	3rd Generation
3GPP	3rd Generation Partnership Project
AAC	Advanced Audio Coding
ACELP	Algebraic Code Excited Linear Prediction
A/D	Analog to Digital
ALSA	Advanced Linux Sound Architecture
AMR	Adaptive Multi-Rate
AMR-NB	AMR Narrow-Band
AMR-WB	AMR Wide-Band
AP	Access Point
ARM	Acorn RISC Machine
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CELP	Code Excited Linear Prediction
CISC	Complex Instruction Set Computer
CN	Comfort Noise
CNG	CN Generation
codec	coder / decoder
CPU	Central Processing Unit
D/A	Digital to Analog
DARAM	Dual Access RAM
DCT	Discrete Cosine Transform
DMA	Direct Memory Access
DSP	Digital Signal Processor
DTX	Discontinuous Transmission
ETSI	European Telecommunications Standards Institute
EVM	Evaluation Module
FEC	Forward Error Correction
FFT	Fast Fourier Transform

FIR	Finite Impulse Response
FPS	Frames Per Second
FTP	File Transfer Protocol
GCC	GNU C Compiler
GPOS	General Purpose Operating System
GPP	General Purpose Processor
GPRS	General Packet Radio Service
GSM	General System for Mobile Communications
HTTP	Hypertext Transfer Protocol
I2C	Inter-IC
IC	Integrated Circuit
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
I/O	Input / Output
IP	Internet Protocol
IPBUF	Interprocessor Buffer
ISO	International Organization for Standardization
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Sector
LAN	Local Area Network
LP	Linear Prediction
LPF	Low Pass Filter
MAC	Multiply And Accumulate
MFLOPS	Millions Floating Point Operations Per Second
MIME	Multipurpose Internet Mail Extensions
MIPS	Millions Instructions Per Second
MITA	Mobile Internet Technical Architecture
MMC	Multimedia Card
MMU	Memory Management Unit
MOS	Mean Opinion Score
MP3	MPEG 1 Layer 3
MPU	Microprocessor Unit
MPEG	Moving Picture Experts Group
MSS	Media Subsystem
NFS	Network File System

OMAP	Open Multimedia Applications Platform
OS	Operating System
OSI	Open Systems Interconnection
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
PDROM	Program and Data ROM
PESQ	Perceptual Evaluation of Speech Quality
POSIX	Portable Operating System Interface
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RTOS	Real-Time Operating System
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SARAM	Single Access RAM
SD	Secure Digital
SDP	Session Description Protocol
SDRAM	Synchronous Dynamic RAM
SID	Silence Descriptor
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
TID	Task ID
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VAD	Voice Activity Detector
VCR	Video Cassette Recorder
Vocoder	Voice Coder
VoIP	Voice over IP
WAV	Waveform Audio Format
WG	Working Group
Wi-Fi	Wireless Fidelity
WLAN	Wireless LAN

List of Figures

List of Tables

Chapter 1

Introduction

In the past, homes were connected to Internet through circuit switched telephone networks with slow connection speeds. Nowadays telephony services are more and more build on top of Internet. Third generation mobile networks are one example of this. Conventional phone calls are transformed from circuit switched systems to packet networks. Internet telephony has become a commodity because of services like Skype [70] and Google Talk [18].

Mobile phones with Wi-Fi (Wireless Fidelity) interface provide means to establish low-cost or free phone calls. The development is in a turbulent phase with many different services and networks and no unbroken or cohesive systems exist, yet.

This thesis seeks to enlighten Internet telephony challenges and requirements by introducing a solution to combine a Voice over Internet Protocol (VoIP) application with 3rd generation speech coding mechanism and a state-of-the-art mobile Linux terminal.

Two patent applications([33], [34]) were filed by the author during the writing of this thesis.

1.1 Motivation

The motivation for this thesis was in the observations in the convergence from traditional phone services and Internet multimedia to handheld computers and mobile phones. More complex requirements are set for communication and related services.

Borders of Internet and operator networks are already fuzzy for the consumers at least. A handheld user who can route a call through a wireless access point to a GSM phone abroad is unaware of different networks utilized. Also 3rd generation (and beyond) mobile networks can be considered parts of Internet due to their packet based characteristics.

1.2 Scope

The scope of this thesis is in the study of speech coding mechanisms in IP networks and in integration of a specific speech coder to a digital signal processor (DSP) utilized by an IP Telephony application. The focus is in standardized speech coding whereas proprietary services, musical audio coding, video and real-time issues of operating systems are demarcated.

This is done in order to achieve proper introduction to current technologies and to provide extensive background for the actual implementation presented in Chapter 6. This should result in gaining leverage using DSP in VoIP applications.

1.3 Goal

The goal of the thesis is to provide an architecture where a VoIP application using complex speech coder can be executed in real-time in a mobile terminal. The architecture needs to be build partly on top of a General Purpose Operating System (GPOS) and partly on a DSP to encode and decode speech.

1.4 Organization of the Thesis

This thesis is divided in chapters. The overview sections in each chapter provide a granny-proof approach for the topic under study. A more in-depth discussion is given in subsequent sections.

First, in Chapter 2 we go through the best standardized practises in VoIP and media streaming on the Internet, followed by different speech coding schemes introduced in Chapter 3, and utilization of DSPs in Chapter 4. The latter chapters present the chosen and designed architecture (Chapter 5), the implementation of the architecture (Chapter 6) and the results (Chapter 7). Final words are stored in Chapter 8.

Chapter 2

Media Transport in IP Networks

2.1 Overview

This chapter introduces the evolution of mobile networks and terminals to IP-based (Internet Protocol) networking. The topics covered include different mobile networks and the protocols utilized in real-time and media streaming services run in IP networks. The end-to-end media negotiation between different terminals, media processing and voice quality issues are also discussed.

Over the years IP Telephony, or Voice over IP (VoIP), has become an option for telephony over circuit switched networks. Many telephone operators offer IP Telephony services [71] or route their traffic through IP networks. VoIP calls are a growing custom among home users. The mobile domain is also moving toward IP-based networking.

It is assumed that the reader is familiar with the network organization in general and has an understanding of the structure of OSI (Open Systems Interconnection) Reference Model network layers [9].

2.2 Current Telecom and Mobile Networks

To better understand the nature and requirements of media transport in mobile environment, current mobile networks are briefly discussed. Long distance operator networks and short distance proximity networks are introduced.

2.2.1 Second and Third Generation

The first mobile networks were built by the radio amateurs which provided base stations (or repeaters) to enable longer distances between the communicating parties. First generation commercial mobile networks were built in the early 1980s and supported traditional voice

services. Second generation networks were dominated by Groupe Spécial Mobile, or Global System for Mobile Communications (GSM), the first commercial digital mobile network. The first version of GSM supported circuit switched voice and data services (GSM Data) in low bit rates. General Packet Radio Service (GPRS) was developed to enhance GSM with packet switched data services. GPRS enables data connections with significantly faster initialization times than GSM Data. The GSM and GPRS networks are together referred to as 2.5G networks. [46], [19]

EDGE (Enhanced Data for GSM Evolution) is based on the same principles as GPRS, but is leveraged by using a more flexible error correction scheme over-the-air. This leads to higher average bandwidth, but variations in bandwidth are increased. The evolution from GSM over GPRS to EDGE can be deployed in the same system infrastructure. Upgrades to hardware and software are necessary, but the overall system architecture remains the same. [19]

Universal Mobile Telecommunication System (UMTS) is a standardized, third generation wireless network, resulted from the GSM/GPRS evolution. It is standardized by Third Generation Partnership Project (3GPP) [5]. UMTS offers data transfer rates up to 2 Mbps and supports a number of multimedia and enhanced telephony services.

2.2.2 Wireless Local Area Networks

A wireless local-area network (WLAN) is a radio frequency network without the limitations of wires or cables. It provides the features and benefits of traditional LAN technologies such as the Ethernet. WLANs have been deployed to office and campus areas since the late 1980s to add mobility to the networking environment.

The connection between the mobile client and the wired network is made through an Access Point (AP). A wireless access point combines routing and bridging functions: it bridges network traffic, usually between wired network over radio interface to computers with wireless adapters. The AP is attached to a node connected to the wired network and acts as a gateway for wireless data to be routed onto the wired network. Like a cellular network the WLAN is capable of roaming between the access points; reconnecting to the network through another access point. [48]

The WLAN network architecture was standardized after the first commercial implementations had existed for some years. The first standard (IEEE-802.11a) was developed by IEEE working group 802.11 and was accepted by the IEEE board during 1997. Other standards include HomeRF and HomeRF 2.0, IEEE-802.11b,c,g and HiperLAN. [22], [48], [36]

The 802.11 networks use unlicensed frequency band around 2.4 GHz. This band has no requirements for end-user licenses while the transmitted power is below 1 W and the transmitter uses a special spread spectrum transmission technique. The 1 W power restriction

serves to limit the range where two radios may interfere with each other. The spread spectrum requirement is intended to make the WLAN signal appear as background noise to a narrowband receiver. [48]

2.2.3 Convergence of Telephone and IP Networks

The trend of creating more data services has prepared the way for moving voice services from PSTN (Public Switched Telephone Network) to packet networks. A converged network incorporates both telephony and data services. The wireless industry has adopted the trend of moving toward converged networks and the services are built over the IP layer. Some of the reasons are discussed below.

A converged network removes the need for a circuit-switched core network. Current studies show that convergence results in lower costs and expenditures for companies' network architectures compared to separate telecom and data networks. The costs for maintaining two separate networks can be reduced in terms of simplification of the network and smaller operational staff when the single network performs both data and telephony services. The integration also eases the offering of enhanced multimedia services. A single standard based network also allows rapid deployment of new services. Operators could gain cost savings by using IP network components over telephony components due to competition and open standards. [72], [17], [46], [7]

IP Multimedia Subsystem (IMS), standardized within 3GPP, is a good example of a comprehensive telephony architecture built over the IP network. IMS provides a service platform for telephone services and real-time multimedia services that easily integrate with each other. By bringing together call control and service provisioning into a horizontally integrated system, IMS enables new combinations of services and service elements, like instant messaging and user presence.

IMS services are based on the Session Initiation Protocol (SIP) (see Section 2.3.3). SIP services are specified to be independent of the access media. This principle of access independence means that IMS services can be accessed over current telecom packet networks, WLAN and wired Internet.

2.3 Media Transport and Control

To gain knowledge of the requirements and the environment of real-time media applications we go through transport, signaling and media negotiation protocols currently utilized in the IP networks. These topics also serve as clarification for the implemented architecture described in Chapter 5. The more detailed studies of media streaming and respective protocols are presented in [68], [39].

The Internet media transport and signaling protocols are specified in Internet Engineering Task Force (IETF) Working Groups (WGs). Instead of abandoned telecom approach to media communications, standardization is led by individuals of university research teams and mobile companies. For example, the International Telecommunications Union Telecommunications Sector (ITU-T) specified media initiation and negotiation protocol H.323 [25] was deployed only in small scale and is replaced in practice by Session Initiation Protocol (SIP), discussed in Section 2.3.3. The common denominator for each IETF specified protocol is the text-based syntax of the protocol. This helps in parsing and debugging and thus enables human readability.

2.3.1 Real-Time Transport Protocol

Real-Time Transport Protocol (RTP) [64] is the only standardized protocol for media transport on the Internet. It provides a common layer for media transport that is independent of signaling protocols and applications. RTP is a result of the standardization work done by Audio/Video Transport (AVT) Working Group of the IETF. The first version of the protocol was published in 1996 and the new revised version [65] was completed in the mid 2003. International Telecommunications Union (ITU) has also adopted RTP as the media transport protocol for H.323 recommendations. [25]

RTP offers a way to deliver digitized audio and video securely across an IP network by incorporating different profiles and payload formats and a number of services for real-time media. These services include loss detection and correction, timing recovery, payload and source identification, reception quality feedback, media synchronization and membership management [52]. Nevertheless, RTP does not take Quality of Service (QoS) issues into account, but these are a subject for consideration in lower level layers of OSI (Open Systems Interconnection) Reference Model [9]. RTP can be utilized over any transport layer, but UDP provides a minimal set of extensions to IP and, thus, low overhead to the RTP packet. UDP is the common choice for an implementation.

RTP is divided in two parts, data transfer protocol and associated control protocol. Real-time Transport Control Protocol (RTCP) provides RTP stream information for stream management. This information contains periodic reporting of reception quality, participant identification and other information describing the RTP stream sender, notification on changes in session membership and the information needed to synchronize media streams [52].

In RTP each stream is transported via a separate RTP session. Due to different media coding methods and delays in packet transmission the streams will go non-synchronous. This effect is commonly noticeable in case of non-synchronous movement of the speaker's lips and produced speech. The solution to the phenomenon is called lip synchronization. RTP provides the information needed for synchronization of multiple media streams.

The reasons for avoiding the bundling of different media streams together are the result of the network limitations, codecs and application requirements [52]. Some participants of a video conferencing session may prefer audio over video, while others have resources for audio only. Participants with both audio and video capabilities may allow lower Frames Per Second (FPS) rates for the video but set higher requirements for the audio quality. In the transport layer this could appear as reservations with differing Quality of Service (QoS) classes for audio and video. [1]

Jitter Buffering

A play buffering is generally used for minimizing errors; a generally known feature in portable "jog proof" CD players. The size of this jitter [2] buffer determines the delay. A smaller buffer decreases the delay but packet loss is increased; if packets are received in disorder the small buffer might cause the application to drop the missing packet. Thus, by increasing the jitter buffer disadvantages of missing packets to media quality can be avoided to some extent. The trade-off between the quality (loss of packets) and minimum delay is found by the requirements and demands of the application. Interactive or conversational real-time applications are more delay-sensitive and set more strict limitations to buffer size. One-way real-time streaming allows more delay and thus larger buffers can be utilized to enhance the transmission quality. The following Sections 2.3.2 and 2.3.3 describe the main signaling protocols for RTP sessions. The example [81] below describes the implications of a jitter buffer.

> Source A is sending audio stream to receiver B. When the delay is fixed, for example 100 ms for every packet, then A continuously sends packets, for example every 20 ms, and B continuously receives packets every 20 ms. Due to fixed delay the inter-arrival time of packets is constant at the receiving end B. In this case no buffering is needed.
>
> In case of variable delay things get more complicated. When A sends packets at interval of 20 ms, due to the variable network delay, B receives them at relative time of 0, 20, 45, 70, 80, 107, ... The variation in delay is 0, 0, 5, 10, 0, 7, respectively. If B is playing packets back every 20 ms and the first packet was played back at the time 0, then when the third packet is supposed to be played it has not yet arrived (at the time of 40 ms). Logically the packet will

[1] In a best-effort network the situation is realized by adding more error correction to the media stream. The solutions include fragmentation of packets, increasing re-sending rate or decreasing packet size. These usually increase the need for the bandwidth and may cause cumulative network congestion. Bundling several streams together would even increase the required bandwidth or leave some media superfluous.

[2] Jitter is defined as the variation in delay.

be skipped. Similarly the fourth packet has not arrived at the time of 60 ms and is skipped, too. This occurrence can be solved by keeping the playback delay at 10 ms, for example. Instead of playing the packets at 0, 20, 40, 60, 80, 100, they are played at 10, 30, 50, 70, 90, 110.

Higher jitter can be absorbed with higher playback delay, but that introduces more overall delay to the end user. If interactive end-to-end latency or say, conversational delay is more than 250 ms it is perceivable and annoying [50]. On the other hand, if the packet can not be played due to late arrival, it is effectively lost. Clearly, there is a trade-off here between the delay and the packet loss.

RTP Payload Format

The RTP framework for the payload formats defines how particular media types are transported within RTP. Payload formats are referenced by RTP profiles[3], and they may also define certain properties of the RTP data transfer protocol. [52]

The relation between an RTP payload format and profile is primarily one of a namespace, although the profile may also specify some general behavior for payload formats. The namespace relates the payload type identifier in the RTP packets to the payload format specifications allowing an application to relate the data to a particular media codec. In some cases the mapping between payload type and payload format is static; in others the mapping is dynamic via an out-of-band control protocol. For example, the RTP profile for Audio and Video Conferences with Minimal Control [63] defines a set of static payload type assignments, and a mechanism for mapping between a MIME (Multipurpose Internet Mail Extensions) [14], [15] type identifying a payload format, and a payload type identifier using the Session Description Protocol (SDP). SDP is discussed in Section 2.3.4. [52]

The relation between a payload format and the RTP data transfer protocol is twofold: a payload format will specify the use of certain RTP header fields, and it may define an additional payload header. The output produced by a media codec is translated into a series of RTP data packets – some parts mapping onto the RTP header, some into a payload header, and most into the payload data. The complexity of this mapping process depends on the design of the codec and on the degree of error resilience required. In some cases the mapping is simple; in others it is more complex. [52]

At its simplest, a payload format defines only the mapping between media clock and RTP timestamp, and mandates that each frame of codec output is placed directly into an

[3] In general, RTP profile defines rules for mapping codec specific information to RTP headers. At the time of writing this thesis there is a single RTP profile: the RTP Profile for Audio and Video Conferences with Minimal Control. [63]

RTP packet for transport. However, this is not sufficient in many cases. Many codecs were developed for circuit-switched systems and without reference to the needs of a packet delivery system. These codecs need to be adapted to this environment. Others were designed for packet networks but require additional header information. In these cases the payload format specification defines an additional payload header and rules for generation of that header. [52]

There are also payload formats that specify error correction schemes. For example, RFC 2198[51] defines an audio redundancy encoding scheme and RFC 2733 [57] defines a generic Forward Error Correction (FEC) scheme. In these payload formats there is an additional layer of indirection, the codec output is mapped onto RTP packets and those packets themselves are mapped to produce an error-resilient transport. To enlighten the nature and the requirements of the implementation part of this thesis, Adaptive Multi-Rate (AMR) speech codec and its RTP payload format are discussed in Sections 3.3 and 3.3.1. [52]

2.3.2 Real-Time Streaming Protocol

Real-Time Streaming Protocol (RTSP) is a standardized media transport control protocol for client-server interaction. It provides a control mechanism for streaming content, e.g. video-on-demand services. RTSP was developed by Multiparty Multimedia Session Control (MMUSIC) Working Group of the IETF and published as a proposed standard in April 1998 [66]. Development of the protocol is proceeding and a revised version will be published in the future [67].

As in the case of RTP, the control channel of RTSP is specified to be independent of the underlying transport layer. The most common implementations utilize both TCP and UDP. The control mechanism is also kept independent of the media part, but RTP is the de facto implementation for media transport.

RTSP is a HTTP-based request-response protocol, incorporating HTTP message format with similar representation of headers and payload. RTSP provides a set of methods for initiating, establishing, playing, recording and a VCR (Video Cassette Recorder) style of controlling the streaming session (presentation). The RTSP methods[4] are listed in Table 2.1.

The RTSP session can enclose several media streams. A typical set could include one stream for audio and one for video. The synchronization of the media streams (lip synchronization, for example) is a media transport or an application layer issue.

Figure 2.1 depicts an example of a basic RTSP session with one RTP media stream. First, the RTSP client retrieves the session description from the RTSP server by sending a

[4] At the time of writing this thesis it is still unclear whether or not the RECORD method should be excluded from the revised RTSP standard. It has been realized that only few practical use cases exist for RECORD and this results in excluding the implementation of the method in most streaming applications.

Table 2.1: The Real-Time Streaming Protocol (RTSP) methods. RTSP provides a set of methods for controlling the media presentation. The left-hand side of the table shows the method name followed by the explanation on the right.

Method	Explanation
DESCRIBE	Retrieves the description of a presentation or media object identified by the request URL from a server.
GET_PARAMETER	Request retrieves the value of a parameter of a presentation or stream specified in the URI.
OPTIONS	Represents a request for information about the communication options available on the request/response chain identified by the Request-URI.
PAUSE	Temporarily halts a stream without freeing server resources.
PING	Prevents the identified session from being timed out.
PLAY	Starts data transmission from the server to client on a stream allocated via SETUP.
RECORD	Starts data transmission from the client to server on a stream allocated via SETUP.
REDIRECT	Indicates that the session should be moved to new server location
SET_PARAMETER	Requests to set the value of a parameter for a presentation or stream specified by the URI.
SETUP	Causes the server to allocate resources for a stream and create a RTSP session.
TEARDOWN	Frees resources associated with the stream. The RTSP session ceases to exist on the server.

DESCRIBE message to the server. In SETUP phase a state for a media stream is reserved. The first SETUP also causes the server to reserve resources for the new RTSP session created. The PLAY message requests the server to start sending media stream enabled in SETUP phase. TEARDOWN message ends the presentation and releases the resources reserved at the server.

2.3.3 Session Initiation Protocol

The Session Initiation Protocol (SIP) is a standardized application layer signaling protocol developed for setting up multimedia calls over IP networks and is mainly used on the Internet. SIP was originally developed by the MMUSIC Working Group of IETF in 1997. The protocol evolved to version 2.0 the next year and after a number of updates and bug fixes the new RFC was published in June 2002 [59].

Many telephone operators already offer SIP services in interconnection with regular telephone services [62]. SIP has been chosen by 3GPP for signaling and establishing traditional

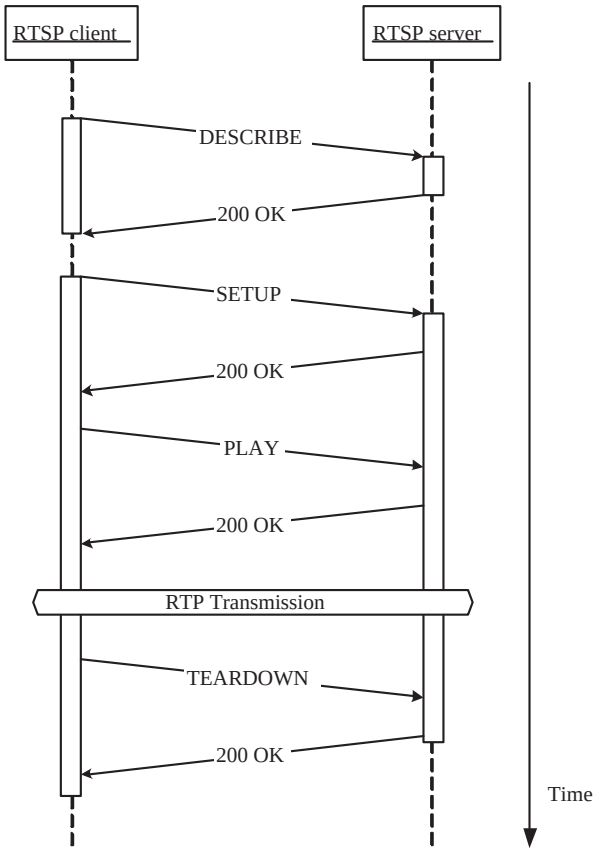

Figure 2.1: Media Stream Signaling with Real-Time Streaming Protocol (RTSP). The figure depicts the establishment and streaming of an RTSP presentation. The actual media transmission is carried out by RTP. First, the media content parameters are requested with stateless DESCRIBE. The following requests create a state (vertical white bars) for the RTSP session that is destroyed after TEARDOWN message.

telephony and enhanced multimedia sessions in UMTS Release 5 [1] and Release 6 [4] networks. A comprehensive study of SIP operations in UMTS IP Multimedia Subsystem (IMS) is also discussed in [45], [46].

The key features of SIP are its flexibility and scalability. It can be used in PC's, laptops, telephone switches, gateways, wireless devices and mobile phones. SIP incorporates elements of two fundamental Internet protocols: HTTP (Hypertext Transfer Protocol) used browsing the web and SMTP (Simple Mail Transfer Protocol) used for e-mail.

Whereas RTSP provides control over the delivery of streaming media, SIP was developed for the needs of IP Telephony. Due to their common nature of media control and signaling (and construction site at MMUSIC) they share many similarities in terms of protocol syntax,

headers and payload. Syntax of both protocols is HTTP-based and they utilize RTP as the de facto protocol for media transport.

SIP protocol complies to request-response model as of other HTTP-based protocols. A SIP session is controlled with requests, and responses for these are provided by the other end it being a server or a client. As a response to a request the receiving party sends a SIP response which gives a response code indicating how the request was processed. Response codes are divided into six categories depending on the general form of behavior expected. These categories are presented below in Table 2.2.

Table 2.2: Session Initiation Protocol (SIP) Methods and Response Classes. The Table first lists the SIP methods. Second, response classes with their numeric codes are presented.

Request Method	Purpose	
INVITE	Invite the user or a service into a session.	
OPTIONS	Discover the capabilities of the receiver.	
BYE	Terminate (hangup) a call or call request	
CANCEL	Terminate incomplete call requests	
ACK	Acknowledge the final response to INVITE.	
REGISTER	Register the current location of a user to the user's home network	
Response Class	**Meaning**	**Example**
1XX	Information about call status. Request received, continuing to process the request.	180 RINGING
2XX	Success	200 OK
3XX	Redirection to another server	301 MOVED TEMPORARILY
4XX	Client did something wrong	401 UNAUTHORIZED
5XX	Server did something wrong	500 INTERNAL SERVER ERROR
6XX	Global failure	606 NOT ACCEPTABLE

A study of combining the features of RTSP and SIP was carried out for this thesis. As a result, a solution was introduced that enables the RTSP functionality, including media control, on SIP with minor extensions (patent [33] pending). For example, RTSP lacks the support for media negotiation that is a basic feature of SIP. By utilizing existing features of SIP the streaming session could be enabled with RTSP inherited headers. VCR style control (Fast Forward, Pause, etc.) was made possible with SIP INVITE. This solution is now introduced as an option in SIP-RTSP interworking in IETF MMUSIC Working Group. [33], [84]

Figure 2.2 depicts a streaming session controlled with SIP. First the media for the session is negotiated. The session will start as RTP audio only. After the server comes aware of

appended video feed it informs the client by sending a re-INVITE. The client has also video capabilities and the streaming session continues seamlessly with both RTP video and audio streams.

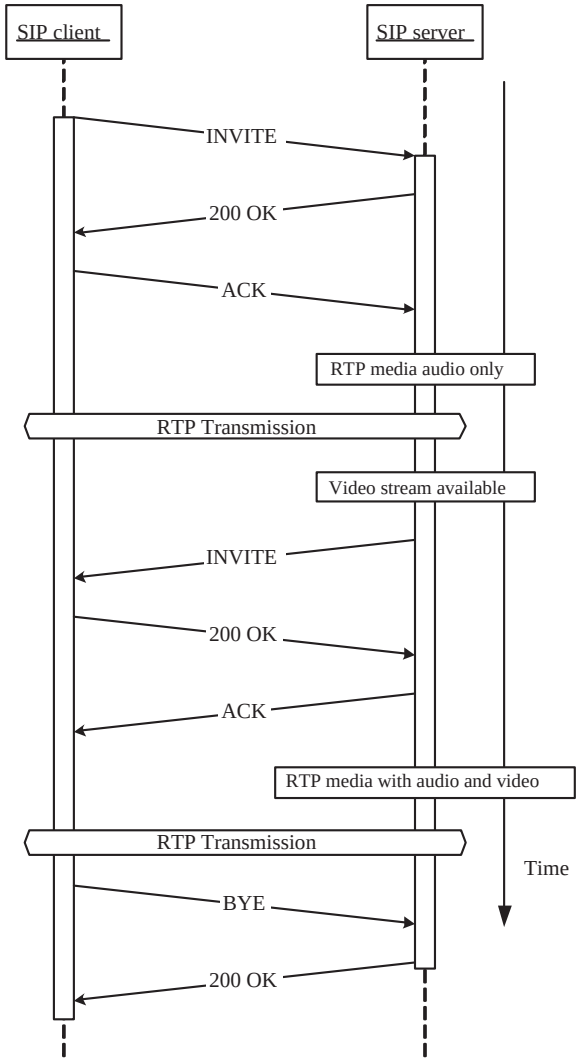

Figure 2.2: Session Initiation Protocol (SIP) with Media Extensions. SIP is extended with minor changes to support RTSP-like media streaming control. The SIP media negotiation (adding the additional video stream during the session) that RTSP is incapable of is also utilized.

2.3.4 Media Negotiation

Establishment of IP Telephony calls or streaming media sessions requires participants to be aware of details of media streams (like codecs and bandwidth), transport addresses and other session description metadata. Session Description Protocol (SDP) [20] was developed in IETF Multiparty Multimedia Session Control (MMUSIC) Working Group around 1998 to provide a standard representation for such information.

SDP can also be utilized in a so called offer-answer model [58]. Offer-answer refers to media negotiation where both ends of the multimedia session announce their media coding capabilities. This is utilized in Session Initiation Protocol (SIP) (see Section 2.3.3) where the SDP is enclosed in `INVITE` request of the initiating party. The receiving end compares local and remote capabilities and sends the section of both back to the initiator in the `200 OK` response payload.

The development of the protocol is ongoing and a revised standard [21] is to be finished in the future. The protocol is independent of how the media is transported and of transport protocols. SDP is intended to be carried on with different transport protocols as appropriate, including SIP, Real-Time Streaming Protocol (RTSP), electronic mail using the Multipurpose Internet Mail Extensions (MIME) [14], [15], and the Hypertext Transport Protocol (HTTP) [12].

As other IETF specified protocols, SDP syntax is also text-based. An SDP session description consists of consecutive lines in format `<key>=<value>`. Session description is divided in session level descriptions and optionally media-level descriptions. Session-level descriptions apply to media level descriptions unless they are defined in the media description. The session level description starts with "`v=`" line and continues to the first media level section, which starts with a "`m=`" line. There can be several media level sections. Table 2.3 lists the SDP fields with their explanations.

Table 2.3: The Session Description Protocol (SDP) fields. The left-hand side of the table depicts the field keys followed by the explanation on the right.

Session description	
a	Session attribute. (opt)
b	Bandwidth information. (opt)
c	Connection information. (opt) Not required if included in all media.
e	Email address. (opt)
i	Session information. (opt)
k	Encryption key. (opt)
o	Owner/creator and session identifier.
p	Phone number. (opt)
s	Session name.
u	URI of description. (opt)
v	Protocol version.
z	Time zone adjustments.
Time description	
t	Time the session is active.
r	Repeat times. (opt)
Media description	
a	Session attribute. (opt)
b	Bandwidth information. (opt)
c	Connection information. Optional if included in the session description.
i	Media title. (opt)
k	Encryption key. (opt)
m	Media name and transport address.

The example below shows a media configuration that is described in SDP as follows.

```
o=user 1189641421 2 IN IP4 10.3.2.2
s=-
c=IN IP4 10.3.2.2
t=0 0
m=audio 29680 RTP/AVP 96 8 0
a=rtpmap:96 AMR/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
```

The `o=` line announces the session owner (`user`) with a generated session id. The following data represent the IP protocol version and the host's IP address. The media address is the same as IP address of Session creator. The following `m=` line informs that only audio is used with media port number being `29680`. The RTP profile in use is AVP (Profile for

Audio and Video Conferences with Minimal Control). The consecutive fields describe the payload types of codecs available. The `a` lines describe the media codecs available in prioritized order, the preferred codec being AMR with sampling rate of 8000 Hz. The other two codecs are PCMA (G.711a) and PCMU (G.711u).

Chapter 3

Speech Coding in Packet Networks

3.1 Overview

The basic function of speech and audio coding is to transform and compress analog audio signal to a digital representation. The advantage of such a transformation is that the manipulation and post-processing of digital audio is more effective than analog audio. Also the transmission of digitized audio is more reliable and the quality can be maintained even with low bandwidth operating modes.

Redundancies introduced in speech signals during the human speech production process make it possible to encode speech at very low bit rates. Furthermore, our perceptual hearing system is not equally sensitive to distortions at different frequencies and has a limited dynamic range. Speech coding techniques take advantage of these properties for reducing the bit rate.

The motivation for reducing the bit rates of speech signals is the demand for cost-effective implementation algorithms, effective usage of bandwidth in both wired and wireless networks and to conserve disk space in storing the audio data. Such applications include voice communications, real-time audio streaming, voice mail and archiving.

This chapter discussed the aforementioned topics and introduces a variety of codecs (coder / decoder) used in wireless and mobile networks. The Adaptive Multi-Rate (AMR) speech coding scheme is the official speech coding scheme for 3GPP (3rd Generation Partnership Project) networks. Three AMR classes exist: both AMR-NB (Narrow-Band) and AMR-WB (Wide-Band) are designed for speech coding. The AMR-WB codec has reasonable performance for music, but it is not comparable to generic audio codecs. AMR-WB+ (Extended AMR-WB) contains both bit-exact AMR-WB and new modes which enable high-quality musical audio streaming. AMR-NB is later discussed in more detail, because it was chosen for the implementation part of this thesis. The implementation work

is discussed in Chapter 6.

3.2 Speech Coding

Figure 3.1 below depicts an abstraction of the end-to-end communications path. First the incoming speech is digitized followed by the parameter estimation. The speech model parameters are then quantized and coded with a specific encoder. Finally, the data is transmitted over the communication channel. At the receiving end, the received data is decoded and reconstructed, and eventually synthesized to speech.

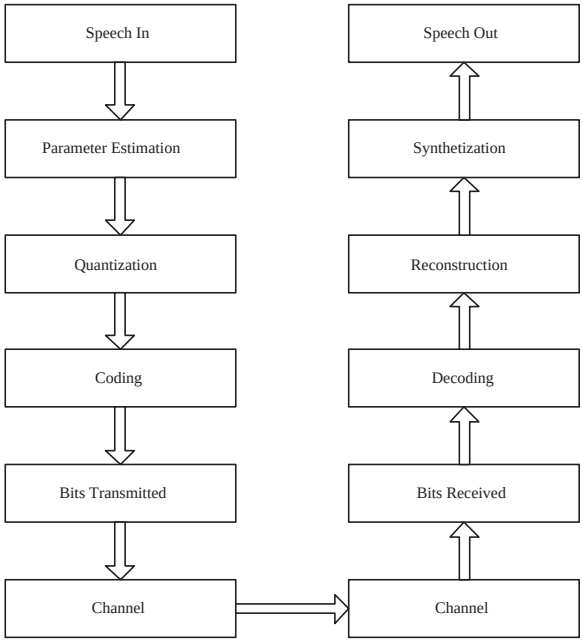

Figure 3.1: Abstraction of End-to-End Speech Processing. Figure depicts the speech processing path from the transmitter to receiver over a communications channel.

Requirement of a narrow bandwidth for speech signals is a bottleneck for signal quality. This obstacle has been tried to bypass by using statistical approaches, for example. A method introduced by P. Jax and P. Vary [28] uses a codebook to extend the coded speech for narrow-band (telephone, for example) to wide bandwidth. The perceived signal quality is increased without transmitting new information for the receiver. The narrowband input signal is classified into a limited number of speech samples for which the information about the wideband spectral envelope is taken from a pre-trained codebook. By using this single codebook the enhanced speech exhibits a significantly larger bandwidth than the input speech without introducing objectionable artifacts.

For the bandwidth efficient transmission of speech signals compression is necessary, for example in mobile communications. Reconstruction with the best possible subjective quality must be maintained even if the transmission channel introduces perturbation. High quality is attained at low bit rates by exploiting signal redundancy as well as the knowledge that certain types of coding distortion are imperceptible because they are masked by the signal.

The design of a speech coding algorithm is limited by the major requirements of high quality decoded speech signals, low bit rate, low computational complexity, high robustness against channel errors and low algorithmic delay.

For a given application, a trade-off between these contradicting requirements has to be found. In speech or audio communications systems in general, the use of standardized coding algorithms is crucial. Standardization ensures the interoperability between different products. Speech coding research is often related to normalization activities e.g. of CCITT/ITU-T (Comité Consultatif International Télégraphique et Téléphonique / International Telecommunications Union Telecommunications Sector), ETSI (European Telecommunications Standards Institute) or ISO-MPEG (International Organization for Standardization - Moving Picture Experts Group).

In comparison to audio signals, speech signals can be characterized by a rather low analogue bandwidth and by particular model assumptions that may be used during the design of the coding algorithm. In standard communications applications a telephone bandwidth of 0.3 - 3.4 kHz allows a digital representation at a sampling frequency of 8 kHz. Generic audio signals, e.g. music, have a bandwidth of about 15-20 kHz and thus require a sampling frequency of 32 - 48 kHz.

Speech coders attempt to minimize the bit rate[1] for transmission or storage of the signal while maintaining required levels of speech quality[2], complexity[3], and communication delay. [8]

Speech coding techniques can be broadly divided into two classes of waveform coding and vocoding (voice coding). Waveform coding aims at reproducing the speech waveform as faithfully as possible, but vocoders try to preserve only the spectral properties of speech in the encoded signal without reproducing the waveform. The waveform coders are able to produce high-quality speech at relatively high bit rates; vocoders produce intelligible

[1] Bit rate refers to coding efficiency which is expressed in bits per second (bps).

[2] Speech quality is usually evaluated on a five-point scale, known as the mean-opinion score (MOS) scale, in speech quality testing—an average over a large number of speech data, speakers, and listeners. The five points of quality are: bad, poor, fair, good, and excellent. Quality scores of 3.5 or higher generally imply high levels of intelligibility, speaker recognition and naturalness.

[3] The complexity of a coding algorithm is the processing effort required to implement the algorithm, and it is typically measured in terms of arithmetic capability and memory requirement, or equivalently in terms of cost. A large complexity can result in high power consumption in the hardware.

speech at much lower bit rates, but the level of speech quality—in terms of its naturalness for different speakers—is also much lower. [8]

3.2.1 Perceptual Evaluation of Speech Quality

Perceptual Evaluation of Speech Quality (PESQ) is a perceptual quality measurement tool for voice quality in telecommunications. PESQ was specifically developed by ITU to be applicable to end-to-end voice quality testing under real network conditions like VoIP and PSTN. In February 2001 PESQ was officially approved as new ITU-T recommendation P.862. [49],[26]

Figure 3.2 depicts the block diagram of the structure of PESQ algorithm. The most significant result of PESQ measurement is the Mean Opinion Score (MOS) that directly indicates the voice quality. The PESQ MOS as defined by the ITU recommendation P.862 has the range from 1.0 (worst) up to 4.5 (best). The common ITU scale ranges from 1.0 to 5.0, but PESQ simulates a listening test and is optimized to reproduce the average result of all listeners. Statistics however prove that the best average result one can generally expect from a listening test is not 5.0, instead it is around 4.5. It appears the subjects are always cautious to score a 5, meaning "excellent", even if there is no degradation at all. [49]

PESQ time alignment includes support for detecting active speech by using Voice Activity Detection (VAD). VAD is a mechanism for reducing the bit rate during silence periods, and is widely used in several speech codecs. Knowing the individual MOS scores is especially useful for optimizing e.g. comfort noise generation during silence periods or noise reduction systems. [49]

3.2.2 Speech and Audio Quality in Packet Networks

Audio quality is a perceptual and subjective experience and its measures are based on a qualitative set of parameters. In addition to factors of perceptual audio codec (coder/decoder) quality[4] the following related factors need to be considered in packet networks: packet loss, algorithmic and network delay, buffering and multitasking (i.e. task switching schemes) of Operating Systems (OS).

Congestion in the network elements (routers, for example) causes them to discard packets that can not be queued to router buffers. Packet loss causes the received information to be incomplete, but depending of the packet loss and the error correction rate, different coding methods provide adequate means for reconstructing the defective data.

[4] In general, perceptual quality of a decoded audio signal depends on the type of input signal (speech or music), bandwidth and compression rate of the signal.

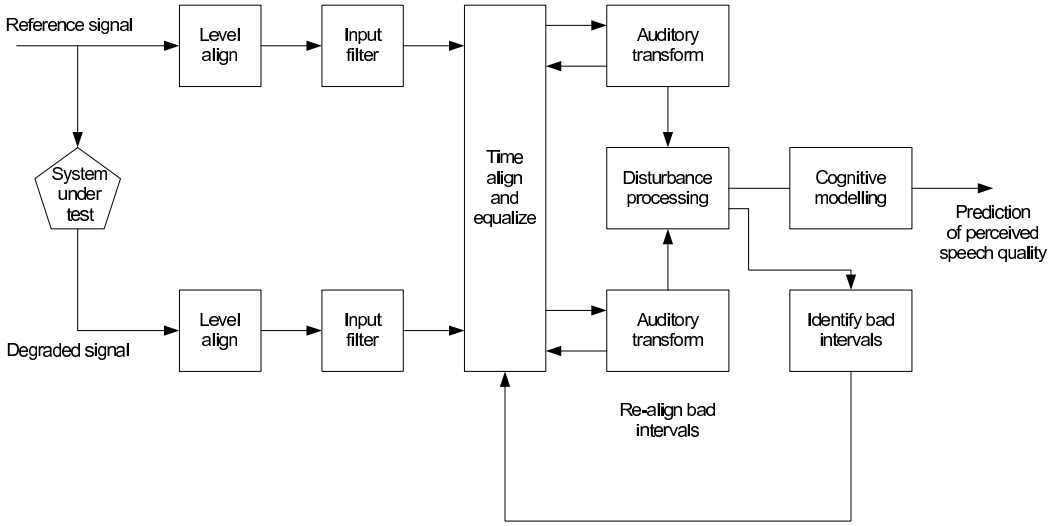

Figure 3.2: Perceptual Evaluation of Speech Quality (PESQ) Algorithm Block Diagram. [49]

Algorithmic delay is caused by the design of the algorithm. If the frame[5] size is constant the delay is constant. With variable frame size the delay varies. However, even if the algorithmic delay is varying, the audio needs to be played out with a constant rate. This naturally limits the minimum playback delay to be the maximum algorithmic delay[6].

Network delay consists of retransmission of dropped packets and the physical limitations of the speed of electromagnetic waves.

Caused by the loss of packets, the order of the incoming packets and variation between inter-arrival times of incoming packets varies. Buffering provides means for post-processing (coding, playing) the data in a constant rate but also causes increased delay. Jitter buffering is discussed in Section 2.3.1.

Multitasking in the context of operating systems causes the media encoding and decoding tasks to suspend for a period of time while the operating system is executing other processes. It is critical for real-time media transport to have media coded in an exactly guaranteed time. Otherwise the media quality might be dramatically reduced. The congestion in media coding buffers causes packet loss as in the case of network elements. An in-depth study of the low latency issues for General Purpose Operating Systems (GPOS) is presented in [82].

[5] A set of samples of arbitrary size. Usually an algorithm has a fixed frame size, for example 20 samples.

[6] Definition of the algorithmic delay also includes the so called look-ahead delay. Look-ahead is caused by the compression algorithm that relies on known voice characteristics to correctly process sample block N. In some algorithms the sample block $N + 1$ needs also to be known in order to reproduce sample block N, which causes additional delay.

3.2.3 Speech Codec Characteristics

The Table 3.1 below lists a subset of audio codecs (coder/decoder) and their features collected in ITU-T Study Group 16 (SG 16) [27]. Codecs that are candidates for the 3rd generation mobile networks are listed below. Table parameters are shortly described in a list below followed by a more in-depth explanation of the complex parameters.

1. Primary Application. A short list of primary applications for the coding standard. The following codes are used:

A	Archival storage
D	Digital Circuit Multiplication Equipment (DCME)
DVD	DVD-video
F	Facsimile
M	Mobile
P	Packet Circuit Multiplication Equipment (PCME)
RN	Radio news
S	Streaming
SVD	Simultaneous voice & data
T	Telephony (general)
TC	Teleconferencing
TV	Television
V	Voice on IP
VC	Video conferencing
VT	Video telephony
W	Wireless LAN
WWW	World Wide Web
A	Approved (list date of first approval date)
D	Draft (list scheduled approval date)
NS	Non-standardized but public (list date of first issue)

2. Nickname: The short, informal name by which the standard is most often referred to.

3. Formal name: The formal identification of the standard - for example, ITU Rec. number, or ISO standard number (not the formal title).

4. Speech Model: Indicates if a speech model is used.

5. Audio Bandwidth: Indicates the range (min - max) of audio passband.

6. Bitrate(s): List of one or more bit rates at which codec can operate. If in format (x-y) this indicates the min - max range.

Table 3.1: Audio Codecs Related to 3GPP. [27]

Nickname	log-PCM	AMR-WB	GSM FR	GSM HR	GSM EFR	AMR-NB	CDMA2000	MPEG-1 Layer III	MPEG-2 AAC	MPEG-4
Formal name	ITU-T G.711	ITU-T G.722.2 (3GPP AMR-WB)	GSM FR (Full-Rate)	GSM HR (Half-Rate)	GSM / PCS 1900 EFR (Enhanced Full-Rate)	3GPP AMR-NB	ANSI	ISO/IEC 11172-3, ITU-R BS.1115	ISO/IEC 13818-7	ISO/IEC 14496-3
Speech Model (Y/N)	N	Y	Y	Y	Y	Y	Y	N	N	Y
Audio Bandwidth (Hz)	300-3400	50-7000	300-3400	300-3400	300-3400	300-3400	300-3400	See sample rates	0-4000, 0-48000	See sample rates
Primary Application (see key)	T	M, S, T, V, VC, SVD	M, V	M, V	M, T, V	M, S, T, V, SVD	N/A	S, V, W	S, V, W	S, V, W, DVD
Bitrate(s) (kbits/s)	48, 56, 64	6.6, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05, 23.85	13	5.6	12.2	4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2, 12.2	0.8, 2.4, 4, 9.6	32 to 320 (8 kbps steps)	Note 7	0.2-300
VAD/DTX/CNG (Y/N, Y/N, Y/N)	N/A	Y/Y/Y	Y/Y/Y	Y/Y/Y	Y/Y/Y	Y/Y/Y	N/A	N/N/N	N/N/N	N/N/N
Frame loss concealment (Y/N)	N/A	Y	Y	Y	Y	Y	N/A	N	N	N
Scalable Bitrate (Y/N)	N	N	N	N	N	N	N/A	N	N	Y
Embedded Scalability (Y/N)	N	N	N	N	N	N	N/A	N	N	N
Sample Rate (kHz)	8	16	8	8	8	8	N/A	32, 44.1, 48	8 to 96	4 to 48
Frame Length (ms)	0.125	20	20	20	20	20	N/A	1152 samples	1024 samples	N/A
Algorithmic Delay (ms)	« 1	25	20	24.4	20	25	N/A	N/A	129.4, 64.7	N/A
Fixed point Comp. Complexity	0.01 MIPS	39.0 WMOPS	2.9 WMOPS	18.5 WMOPS	15.2 WMOPS	16.7 WMOPS	N/A	N/A	1.9-22.8, 3.8-46.2	N/A

7. VAD/DTX/CNG: Voice Activity Detection, Discontinuous Transmission, Comfort Noise Generation

8. Frame loss concealment: Indicates if a codec actively conceals artifacts caused by frame loss.

9. Scalable Bitrate: The encoder has the ability to continuously vary the amount of data sent per unit time, with a granularity of a few bytes or less.

10. Sample Rate: The frequency at which input samples are acquired.

11. Frame Length: The length of each set of independently decodable samples.

12. Algorithmic Delay: The minimum time between acquisition of a given input sample at the encoder and reconstruction of the same output sample at the decoder. This value assumes instantaneous processing and zero propagation delay between encoder and decoder. Usually calculated as frame length + look-ahead buffer [7].

13. Fixed point Computational Complexity: Approximate computational complexity of encoder + decoder. Units vary. This is only a rough approximation, as this value is highly dependent on implementation architecture.

Speech model

Speech model refers to extracting parameters of the sampled speech signal. In vocoders the parameters of a source filter speech model (instead of signal samples) are quantized and transmitted. This source-filter synthesis representation closely resembles the model of speech production.

Frame loss concealment

As discussed in Section 3.2.2, most of the packet-switched networks, like GPRS and Internet, are based on the best-effort principle which does not guarantee Quality of Service (QoS) demands. Packets can be lost in two ways: through dead-end routes or because a router purposely drops packets in order to manage congested links. In addition, speech packets that are delayed too long (i.e. their play out time has already passed) are as good as lost since human conversations cannot tolerate long delays. For example, in backward-adaptive coding schemes (like in G.723.1 and G.729), the packet loss results in loss of synchronization between the encoder and the decoder. This causes the degradations of the

[7] Look-ahead refers to an algorithmic process in which some of the samples from the following frame are used to improve the performance of the compression process.

output speech signal to propagate into following segments of the speech signal until the decoder is resynchronized with the encoder. Codec-specific loss concealment algorithms are used in both encoders and decoders to prevent impeding of lost packets. [61], [60]

Algorithmic delay

Algorithmic delay is delay that is intrinsic to the algorithm of a codec and is independent of CPU speed. The algorithmic delay of a codec is simply referred as the delay of the codec. The algorithmic delay is generally expressed in terms of the number of samples or milliseconds by which a codec's output lags behind the corresponding input. In some cases algorithmic delay is referred as the sum of the basic algorithmic delay and tasking latency of the operating system. The first-mentioned definition is effective in this thesis.

3.3 Adaptive Multi-Rate Narrow Band Speech Codec

The Adaptive Multi-Rate Narrow Band (AMR-NB) speech codec [2] was originally developed and standardized by the ETSI for GSM systems. It was also chosen by 3GPP as the mandatory codec for 3rd generation (3G) cellular systems. Due to its flexibility and robustness, it is also suitable for other real-time speech communication services over packet-switched networks such as the Internet.

AMR-NB consists of the multi-rate speech coder, a source controlled rate scheme including a voice activity detector (VAD) and a comfort noise (CN) generation system during silence periods. The AMR-NB frames containing CN parameters are called Silence Descriptor (SID) frames. It also incorporates an error concealment mechanism to minimize the effects of transmission errors and lost packets.

The multi-rate speech coder is a single integrated speech codec with eight source rates from 4.75 kbits/s to 12.2 kbits/s. AMR-NB features also a low rate background noise mode for reducing the number of transmitted bits and packets during silence periods. The sampling frequency used in AMR-NB is 8000 Hz and the speech encoding is performed on 20 ms speech frames. Therefore, each encoded AMR-NB speech frame represents 160 samples of the original speech. The speech coder is capable of switching its bit-rate every 20 ms speech frame upon command.

A 3GPP reference configuration where the various speech processing functions are identified is depicted in Figures 3.3 and 3.4 for encoder and decoder, respectively. The audio parts including A/D (analog to digital) and D/A (digital to analog) conversion are included to show the complete speech path between the audio input/output and the digital interface of the network. The detailed description of the audio parts is out of the scope of this thesis. Modules of both AMR-NB encoder and decoder are presented in Figures 3.3 and 3.4 are

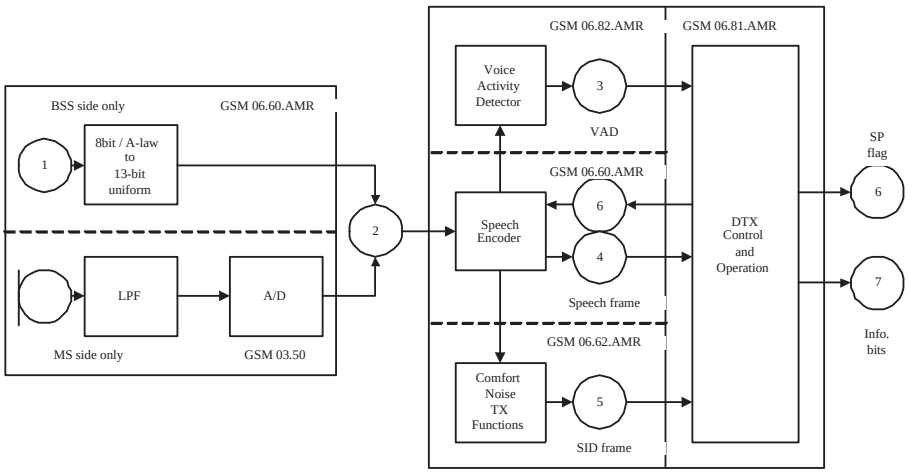

Figure 3.3: Overview of AMR-NB Encoder Audio Processing Functions.

described below in Table 3.2.

As shown in the Figure 3.3, the speech encoder receives its input as a 13-bit uniform Pulse Code Modulated (PCM) signal either from the audio part of the user equipment (mobile phone, for example), or on the network side (BSS, Base Station System) from the Public Switched Telephone Network (PSTN) via an 8-bit A-law or μ-law to 13-bit uniform PCM conversion. The encoded speech at the output of the speech encoder is packetized and delivered to the network interface. In the receive direction, the inverse operations take place. [2]

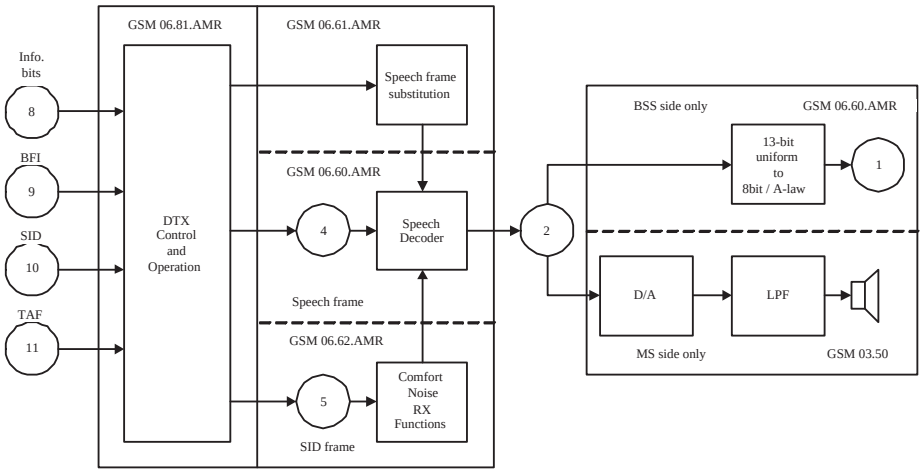

Figure 3.4: Overview of AMR-NB Decoder Audio Processing Functions.

The input block of 160 speech samples is first transformed from 16-bit presentation to

Table 3.2: AMR-NB Encoder and Decoder Modules. The logical AMR-NB encoder and decoder modules are enumerated for the Figures 3.3 and 3.4.

Part	Description
1	8-bit A-law or μ-law PCM (ITU-T Recommendation G.711 [24]), 8 000 samples/s
2	13-bit uniform PCM, 8 000 samples/s
3	Voice Activity Detector (VAD) flag
4	Encoded speech frame, 50 frames/s, number of bits/frame depending on the AMR-NB codec mode
5	Silence Descriptor (SID) frame
6	TX_TYPE, 2 bits, indicates whether information bits are available and if they are speech or SID information
7	Information bits delivered to the 3G AN
8	Information bits received from the 3G AN
9	RX_TYPE, the type of frame received quantized into three bits

13-bit form. This is done by zeroing the three least significant bits. These speech frames are then pre-processed (low-pass filtered, for example) and passed to encoder. The size of encoded block is 118 bytes (59 samples) and the number of non-zero bits depends on the presently used codec mode. The decoder transforms this data to reconstructed speech samples. The coding scheme is Multi-Rate Algebraic Code Excited Linear Prediction (ACELP)[8]. The bit rates of the source codec are listed in Table 3.3. [3], [2]

3.3.1 AMR-NB RTP Payload Format

The AMR-NB Real-time Transport Protocol (RTP) payload format specifies the method for packetization of AMR-NB encoded speech signals into RTP. This payload format supports transmission of multiple channels, multiple frames per payload, the use of fast codec mode adaptation, robustness against packet loss and bit errors, and interoperation with existing AMR-NB and AMR-WB transport formats on non-IP networks. [83]

Even though this payload format specification supports the transport of both AMR-NB and AMR-WB speech data, it is important to remember that AMR-NB and AMR-WB are two different codecs and they are always handled as different payload types in RTP.

[8]ACELP is a speech encoding algorithm employed in analysis by synthesis codecs in order to predict the filter coefficients required to synthesize speech at the receiving party. In the algorithm a limited set of pulses is distributed as excitation to linear prediction filter. The ACELP method is widely employed in current speech coding standards such as AMR, AMR-WB and ITU-T G-series standards G.729 and G.723.1. The main advantage of ACELP is that the algebraic codebook it uses can be made very large (> 50 bits) without running into storage (RAM/ROM) or complexity (CPU time) problems.

Table 3.3: AMR-NB Source Codec Modes and Bit Rates. Each AMR-NB mode name is listed on the left column. The right column represents the actual bit rates of each mode. The abbreviations in the brackets quote the network specific standardized name for the given mode. Particularly, the 6.7 kbps mode is adopted as PDC-EFR [14], the 7.4 kbps mode as IS-641 codec in TDMA [13], and the 12.2 kbps mode as GSM-EFR [12].

Codec Mode	Source Codec Bit Rate
AMR_12.20	12,20 kbits/s (GSM EFR)
AMR_10.20	10,20 kbits/s
AMR_7.95	7,95 kbits/s
AMR_7.40	7,40 kbits/s (IS-641)
AMR_6.70	6,70 kbits/s (PDC-EFR)
AMR_5.90	5,90 kbits/s
AMR_5.15	5,15 kbits/s
AMR_4.75	4,75 kbits/s
AMR_SID	1,80 kbits/s (see note 1)

Chapter 4

Digital Signal Processing in Mobile Terminals

4.1 Overview

Digital Signal Processor (DSP) is a computer, optimized for the detection, processing and generation of real-world signals such as voice, video and music. It is usually implemented in a single chip in dimensions smaller than a stamp. The price range for a DSP varies from one euro to hundreds of euros. Digital Signal Processors reside in every mobile phone, CD-player or a car. These real-life implementations are generally called as embedded systems.

The difference between a DSP and a microprocessor is not obvious. Earlier the separation was based on a DSP hardware integrated algorithm called Multiply and Accumulate (MAC) [1], used counting the inner product in numerous mathematical functions. Nowadays many DSPs have microprocessor functionality on board and many microprocessors provide DSP functionality[6].

Traditionally IP Telephony applications are run in a general purpose microprocessor (GPP) with a general purpose operating system (GPOS). This chapter discusses the usage of Digital Signal Processors as media coding accelerators in mobile terminals. First, DSP requirements and needs for mobile terminals are enlightened. Processor architectures and communication between the DSP and its periphery are studied. The chapter closes in

[1] DSP provides dedicated hardware for Multiply and Accumulate (MAC) operation. Results of the multiplication of two data items (operands) are usually available within one processor clock cycle. In comparison a typical microprocessor carries out its multiply operation by a binary long multiplication process. When it encounters a multiply instruction, a micro-code — an internal sequence of operations — is invoked. This micro-code performs the multiplication as a sequence of shifts and adds on successive clock cycles until the result is complete. The resulting overhead for multiplication on a microprocessor could be approximately 80 processor clock cycles to perform a 16-bit multiplication. [6]

the study of interprocessor communication.

The approach to Digital Signal Processing in this Chapter is based on software design. DSPs are studied as an alternative platform for running IP Telephony media components. The details of hardware design are bypassed.

4.2 Definitions for Real-Time

There are several definitions for real-time, most of them contradictory [37]. POSIX Standard 1003.1 [23] (discussed in Section 4.6.1) defines real-time for operating systems as:

> Real-time in operating systems: the ability of the operating system to provide a required level of service in a bounded response time.

The following definition [37] covers the viewpoint of real-time DSP processing and is accepted here as a sufficiently exact definition.

> In a real-time DSP process, the analyzed (input) and / or generated (output) samples (whether they are grouped together in large segments or processed individually) can be processed (or generated) continuously in the time it takes to input and / or output the same set of samples independent of the processing delay.

An audio DSP example clarifies the definition. The system is not capable of real-time processing of a 2.00 seconds of sound when analyzing it requires 2.01 seconds. If it requires only 1.99 seconds, it is a real-time process.

Let us consider another example related to sampling and demand for real-time. An audio filter requires 10 μs to process one sample. The DSP can therefore filter audio signals with bandwidth up to 100 kHz. With a microprocessor with capability of processing the sample in 200 μs the upper limit for sampling rate would only be 5 kHz. This clearly indicates how the computational processing speed of a microprocessor limits the signal bandwidth.

4.3 Performance Measurement

The microprocessor performance, or speed, is commonly measured in terms of number of instructions or number of operations in a period of time. Million Instructions Per Second, MIPS, is the most common measure. This measure is commensurate only among processors with the same instruction set, because different instruction sets often take different numbers of instructions to execute the same task [85].

MFLOPS, Million Floating-Point Operations Per Second, and MOPS, Million Operations Per Second, describe the number of operations a device can perform in a second.

FLOPS, like MIPS, are not that useful as a benchmark for modern computers because other factors in computer performance are excluded. The factors include interprocessor communication, cache coherence, and the memory hierarchy. [85]

Another measure of DSP systems is called MMACS (Million of Multiply-Accumulates Per Second). This is the number of MAC operations, in millions, that the device can perform in a second. For comparison of DSP devices, this gives inaccurate results because it does not take the needs of an application into account.

To provide a comparative measure of performance across processors and instruction sets, a System Performance Evaluation Cooperative (SPEC) benchmark was developed by a non-profit organization called Standard Performance Evaluation Corporation [73]. SPEC is a set of standard CPU-intensive integer and floating point benchmarks based on real programs [47]. SPEC CPU2000 is a well recognized benchmark suite that measures the performance of the processor, memory and compiler on the tested system.

SPEC CPU2000 is divided into two suites of benchmarks, SPECfp and SPECint. SPECfp measures the processor's floating-point performance and the CPU's interaction with main memory and cache. Audio encoding, certain spreadsheet calculations and 3D games are examples of floating-point applications. SPECint measures the system's integer performance. SPECint is an applicable measure in the category of word processing, file compression, email and database performance. [74]

General purpose processors used in personal computers integrate instructions for both integer and floating-point operations. In the embedded systems this varies. Hardware needs to give effect to only a specific set of tasks in terms of maximizing the system performance and minimizing the power consumption. Most of the handheld devices, or Personal Digital Assistants (PDAs), provide only a processor with integer instructions. Floating-point operations are implemented as library functions, which lack the efficiency and speed of hardware integration.

Usually DSPs provide also a hardware integrated set of mathematically complex algorithms. In the field of media processing, these include FIR filters, Fast Fourier Transform (FFT) and Discrete Cosine Transform (DCT).

4.4 Digital Signal Processor Architectures

This section discusses different DSP architectures and introduces communication schemes based on interrupts and memory access methods. Finally, the eXpressDSP Algorithm Interoperability Standard (XDAIS) is introduced. XDAIS is developed for DSP code generation to define framework between different parts of DSP application development.

For comparison and better understanding of DSP and microcontroller architecture differ-

ences, an introductory listing is presented in Table 4.1. Microcontroller tasks are clearly related to external communication and control with other devices whereas DSP requirements are strongly algorithm execution related. A closer look into DSP requirements is presented in Table 4.2 where the main characteristics are listed.

Table 4.1: Microcontroller and DSP Usage Comparison. The table clearly shows the non-overlapping requirements from microcontroller and DSP based functionalities. [16]

Embedded Processor	System Requirement	Feature	Benefit
Microcontroller	I/O Control	I/O ports with bit-level control	Efficient control of external devices
	Peripheral Communications	Serial ports, UART, I^2C	Hardware support for expansion; external device networking and communications
	Precision control of motors and actuators	Sophisticated timers	Low software overhead
	Quickly resolve complex software program control flow	Conditional jumps, bit test instructions, interrupt priority control	Efficiently implement control oriented algorithms
	Fast response to external events	External interrupts with multiple priority levels	Program control immediately redirected on event occurrence with minimal overhead
	Conversion of sensor data	A/D converters	Hardware support for translation of analog signals
Digital Signal Processor	Software filters	MAC unit, zero overhead loops	Digital filtering in few cycles
	Interface to codecs	High-speed serial ports	Hardware support for translation of analog analog signals
	High data throughput from serial ports	Peripheral DMA	Less wasted cycles fetching data from serial ports
	Fast data access	Harvard architectures and variants	Fast execution of signal processing algorithms

Traditional DSP architecture illustrates the processor architectural evolution from CISC (Complex Instruction Set Computer) to RISC (Reduced Instruction Set Computer). The CISC influence can be seen from the design philosophy of trying to do more with one instruction, which leads to compact code size, and the use of memory operands to increase the data throughput. The RISC characteristic of DSP is reflected in the fixed instruction set and the simplified addressing mode. [69]

The combination of RISC and Single Instruction Multiple Data (SIMD) architectures

Table 4.2: Typical Digital Signal Processing Requirements. [69]

Requirement	Implementation
Arithmetic Operation	
Single cycle MAC	Parallel array multiplier
Conditional Execution	Comparison + other op
Saturation overflow behavior and rounding	Saturation and rounding hardware
Parallel barrel shifting	Shifting + other op
Memory Access	
Single cycle dual access in parallel with arithmetic operation	Harvard structure and dual address generators
Special addressing mode	Circular buffer / bi-reverse addressing hardware
Parallel pointer adjustment	Dedicated address adder
Program Control	
Zero overhead looping	Looping hardware
Efficient call / interrupt	Hardware stack and so on

tries to produce a unified DSP-Microcontroller design. The unified system is particularly suited for embedded devices.

Dual core processors contain two processors that are build into a single box. Usually the other one is a GPP and dedicated to act as a host processor for operating system and applications. The other one is a DSP and communicates with the GPP by a proprietary protocol. Dedicated real-time tasks can be placed on DSP and the other processor is reserved for non-real-time processes.

4.4.1 Interrupts

Interrupt-driven I/O is used by almost all systems for at least some devices. The system employs I/O interrupts to indicate to the processor that an I/O device needs attention. When a device wants to notify the processor that it has completed some operation or needs attention, it causes the processor to be interrupted. It is up to system software to handle the interrupt and proceed with the running of the tasks.

When the I/O operation is interrupt-driven[2], the operating system simply works on the other tasks while data is being read from or written to the device. When the OS recognizes

[2]To deal with the different priorities of the I/O devices, most interrupt mechanisms have several levels of priority. These priorities indicate the order in which the processor should process interrupts. Both internally generated exceptions and I/O interrupts have priorities. Typically, I/O interrupts have lower priority than internal exceptions. There may be multiple I/O interrupt priorities, with high-speed devices associated with the higher priorities. [47]

an interrupt from the device, it reads its status to check for errors. If there are none the OS can supply the next piece of data, for example, by a sequence of memory-mapped writes. When the last byte of an I/O request has been transmitted and the I/O operation is completed, the OS can inform the program. The processor and OS do all the work in this process, accessing the device and memory for each data item transferred. [47]

An I/O interrupt is asynchronous with respect to the instruction execution. That is, the interrupt is not associated with any instruction and does not prevent the instruction completion. The control unit needs only check for a pending I/O interrupt at the time it starts a new instruction. [47]

In addition to the fact that an I/O interrupt has occurred, it is essential to convey further information such as the identity of the device generating the interrupt. Furthermore, the interrupts represent devices that may have different priorities and whose interrupt requests have different urgencies associated with them. [47]

4.4.2 Direct Memory Access

A mechanism was created for off-loading the processor and having the device controller transfer data directly to or from the memory without the intervention of the processor. This mechanism is called Direct Memory Access (DMA). The interrupt mechanism is used by the device to communicate with the processor, but only on completion of the I/O transfer or when an error occurs, rather than generating interrupt per transferred byte. [47]

DMA is implemented with a specialized controller that transfers data between an I/O device and memory independently of the processor. The processor grants the DMA controller to bus master. DMA then directs the reads and writes between itself and memory. Figure 4.1 depicts the interaction between the CPU and DMA controller.

There are two possibilities for the timing of the data transfer from the DMA controller to memory. The controller can cause the processor to halt when it attempts to access data in the same memory bank where the controller is writing. This is the fastest option for the I/O device, but may cause the processor to run slowly because the processor may need to wait until a full block of data is transferred.

In the other case the controller can access memory in memory cycles, which are not used by the particular memory bank into which the DMA controller is writing data. This approach is called as cycle stealing and is commonly used approach. The actual DMA transfer consists of three steps [47]:

1. The processor configures the DMA by supplying the identity of the device, the operation to perform on the device, the memory address that is the source or destination of the data to be transferred and the length of data (number of bytes to transfer).

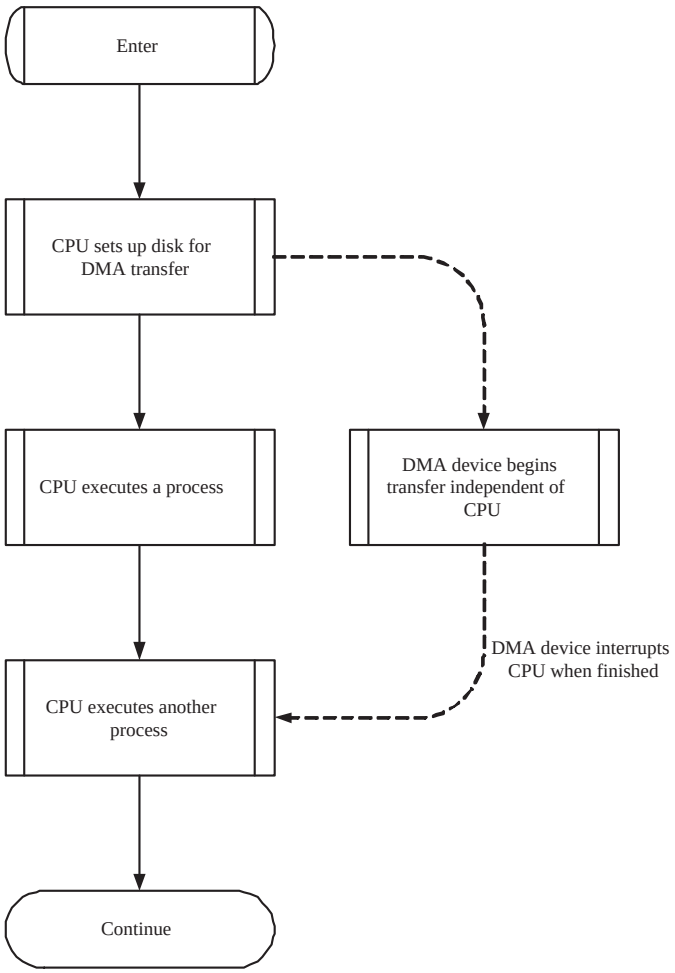

Figure 4.1: Direct Memory Access (DMA) Takes Over the Control of File I/O. Central Processing Unit (CPU) grants DMA to proceed with data transfer while performing other tasks. When finished, DMA interrupts CPU to inform of the completion of data transfer.

2. The DMA starts the operation on the device and arbitrates for the bus. When the data is available (from the device or memory), the data is transferred. The DMA device provides the memory address for the read or write. If the request requires more than one transfer on the bus, the DMA unit generates the next memory address and initiates the next transfer. Using this mechanism, a DMA unit is able to complete an entire transfer, which may be thousands of bytes, without involvement of the processor. Many DMA controllers contain a memory buffer to allow them to deal flexibly with delays either in transfer or those incurred while waiting to become bus master.

3. Once the DMA transfer is complete, the controller interrupts the processor. Processor

then determines by interrogating the DMA device or examining memory whether the entire operation was completed successfully.

There may be multiple DMA devices in a computer system. For example, in a system with a single processor-memory bus and multiple I/O buses, each I/O bus controller will often contain a DMA processor that handles any transfers between a device on the I/O bus and the memory. [47]

4.4.3 XDAIS

The eXpressDSP Algorithm Interoperability Standard (XDAIS) is a Texas Instruments (TI) standard that incorporates an example of software framework which defines a standard set of coding conventions and application programming interfaces (APIs). XDAIS includes algorithm programming rules that enable interoperability between other code blocks.

Algorithms written in the eXpressDSP standard will work on all TI platforms and are easily included in the users software. XDAIS uses interfaces and environment build upon DSP/BIOS II DSP operating system. All DSP Global algorithms are eXpressDSP compliant and have been tested and certified as such by Texas Instruments. [78], [6]

XDAIS defines a set of programming rules and conventions that need to followed for the algorithm to be fully compliant. Most of the rules are common sense programming practices that are widely used and can be applied to all algorithms. In addition to these base rules, there are Instruction Set Architecture (ISA) specific rules. For example, rules that define the usage of specific control registers for each Texas' TMS320 processor family. [6]

4.5 Tasks

This section discusses the task switching between processes and different scheduling algorithms. Processes and threads are followed by introducing context switching mechanisms.

4.5.1 Task Management and Execution

Before running of a DSP task or program it needs to be loaded to the DSP memory. The assembled and linked program can reside in the hard disk or in some other external memory. To load a task the operating system or similar needs to execute the following phases [47].

1. Read the header of executable to determine the size of the text and data segments.

2. Create new address space for the task. This address space includes holds space for text, data segments and stack segment.

3. Copy the instructions and data from the executable file into the new address space.
4. Copy arguments passed to the program onto the stack.
5. Initialize the machine registers. The stack pointer is set to point to the address of the first free stack location.
6. Jump to start-up routine that copies the program's arguments from the stack to registers and calls the program's *main* routine.

4.5.2 Processes

Primitive units for system resources allocation are called processes. Each process is provided with its own address space and (usually) one thread of control. A process executes a program and multiple processes can execute the same program. However, each process has its own copy of the program within its own address space and executes it independently of the other copies. [13]

Processes are organized hierarchically. Each process has a parent process which is explicitly arranged to create one. The processes created by a parent process are called its child processes. A child inherits many of its attributes from the parent process. [13]

Each process has a also unique identifier, process ID number. Process ID is allocated to each process at the time of its creation. The lifetime of a process ends when its termination is reported to its parent process. The termination causes all of the process resources, including its process ID, to be freed. [13]

4.5.3 Threads

According to POSIX Standard IEEE 1003.1-2003, a thread is a single flow of control within a process. Each thread has its own thread ID, scheduling priority and policy, error number value, thread-specific key/value bindings and the required system resources to support a flow of control. Anything whose address may be determined by a thread, including but not limited to static variables, storage obtained via malloc(), directly addressable storage obtained through implementation-defined functions, and automatic variables, are accessible to all threads in the same process. [23]

4.5.4 Task Switching

The task switching between multiple functions can occur between processes or threads. A task priority indicates importance of the task relative to other tasks. It may be fixed or variable, unique or shared with other tasks. A task switch occurs when one task suspends

execution and another starts or resumes execution. Task switch is also called as context switch, because a tasks context (generally the complete contents of the stack and the values of the registers) is usually saved for re-uses when the task resumes. Task switching is also called as scheduling. [53]

Preemption occurs when a task is interrupted by the kernel and another task is prepared to be executable. An alternative to a preemptive system is a cooperative system, in which a task must voluntarily relinquish the control of the processor before another task may start its execution. In this case, the control of task switching is partly left to programmer who needs to structure the task so that this occurs. If a running task fails to cooperate (for example, repeats an infinite loop), other tasks will not execute and the system will fail to work properly [53].

Preemptive and cooperative context switching are handled by the kernel. Kernel software manages the context switching and the communication between processes. The kernel generally ensures that the highest-priority task is the task that is running (preemptive scheduling) or will run next (cooperative scheduling). [53]

4.6 Operating Systems for Signal Processors

An operating system gives advantage for software design when the number and complexity of tasks increase. The OS could be utilized to provide functionality for allocating system resources and to perform multi-tasking.

A number of DSP operating systems exists. Information for Table 4.3 was gathered to list the DSP operating systems and their features in the market. Excluded are the operating systems for standard PC hardware. The features chosen and listed can be considered essential in embedded systems design.

The need for DSP operating system depends on the tasks to be performed. A simple DSP task performing only few functions usually does not leverage of the use an operating system. The operating system would only add overhead to application memory requirements, for example. The programmer can easily provide the means for peripheral and internal communications by using library functions of the code generation software.

The responsibilities of the operating system arise from three characteristics [47] of I/O systems:

1. The I/O system is shared by multiple programs using the processor.

2. I/O systems of the use interrupts (externally generated exceptions) to communicate information about I/O operations.Because interrupts cause a transfer to kernel or supervisor mode, they must be handled by the operating system.

3. The low-level control of an I/O device is complex because it requires managing a set of concurrent events and because the requirements for correct device control are often very detailed.

Many companies offer pre-written operating systems for DSP processors. The operating systems offer several functions absorbed from common operating systems. In addition and most importantly they feature real-time functionality essential in most of the DSP applications.

Features and prices of the operating systems vary. Most companies charge only for the purchase of their code development environment. The environment uses the operating system libraries and integrates the application to the proprietary OS. Charging of some operating systems include also the runtime license fee that needs to payed for each commercial application. Some companies deliver their OS sources for developers to optimize the OS overhead for the application requirements. Operating System can be said to be Real-Time Operating System (RTOS) when the below mentioned conditions are met. [37]

1. An RTOS (Real-Time Operating System) has to be multi-threaded and preemptive.
2. The notion of thread priority has to exist as there is for the moment no deadline driven OS.
3. The OS has to support predictable thread synchronization mechanisms.
4. A system of priority inheritance has to exist.
5. OS Behavior should be known.

This indicates that the following figures should be clearly given by the RTOS manufacturer:

1. The interrupt latency (i.e. time from interrupt to task execution). This has to be compatible with application requirements and has to be predictable. The value depends on the number of simultaneous pending interrupts.
2. For every system call, the maximum time the system call takes. It needs to be predictable and independent of the number of objects in the system.
3. The maximum time the OS and drivers mask the interrupts.

4.6.1 POSIX

Portable Operating System Interface (POSIX) standard is widely implemented in different Operating System environments, including all UNIX platforms. POSIX is an open operating interface standard accepted world-wide. It is produced by the IEEE (Institute for

Electrical and Electronics Engineers), Inc. POSIX standards are also adopted by ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission). [75]

POSIX standard is divided in several parts. The first part was called as IEEE Std 1003.1-1990 or "Part 1: System Application Program Interface (API) [C Language]" defines the C language interface for UNIX-like kernel and its functionality, like process primitives, the environment of a process and file system structure. The second part (IEEE Std 1003.2-1992), "Part 2: Shell and Utilities" defines the shell and about 100 utilities. Later the 1003.1 was updated to include real-time extensions (1003.1b) and threads (1003.1c). This standards (1003.1) is referred as POSIX.1. [75]

The code portability between different operating systems and hardware is assured by implementing POSIX support. The support can be partial or comprehensive. POSIX conformance means that the POSIX.1 standard is supported in its totality. A system that has the conformance certified implements the real-time and threading routines of the POSIX.1b and POSIX.1c subsets in addition to core services. The term POSIX compliance indicates that a system provides partial POSIX support. POSIX compliance means that the product provides also documentation for showing which POSIX features are supported. POSIX, its subsets and services are listed in Table 4.4 below. [32]

Table 4.3: Real-time Operating Systems on the Market. The table lists Real-time Operating Systems available for DSP systems. The information of the Table is gathered from sources [80], [11], [56], [29], [35] [55], [54], [31]. In the Table, pe=preemptive, co=cooperative.

Tool Name	Company	Target	Sources Available	Multitasking	Networking	File System Support	POSIX support	Task Dynamic Linking	Low-power Modes
C EXECUTIVE	JMI Software Systems, Inc.	many	X	pe	TCP/IP, SNMP	DOS	N/A (see PSX)	N/A	N/A
CMX-RTX	CMX Systems Inc.	many	1K - 10K	pe	TCP/IP services	N/A	N/A	N/A	N/A
CMX-Tiny	CMX Systems Inc.	many	1K - 10K	pe	TCP/IP services	N/A	N/A	N/A	N/A
Diamond	3L Ltd.	many	<1K	X	TCP/IP	X	N/A	N/A	N/A
DSP/BIOS II	Texas Instruments	TI	N/A	pe	TCP/IP	N/A	N/A	X	X
KROS	KROS Technologies	Altera	N/A	X	TCP/IP services	FAT16/32, CF	subset of 1b, 1c	N/A	N/A
Nucleus PLUS	Mentor Graphics	many	X	X	TCP/IP services	FAT16/32, PCMCIA / CF	N/A	X	N/A
PSX	JMI Software Systems, Inc.	many	X	X	TCP/IP, SNMP	DOS	subset of POSIX.1	N/A	N/A
RTXC Quadros	Quadros Systems, Inc.	many	N/A	pe	TCP/IP services, IrDA	FAT16/32	N/A	N/A	N/A
Salvo	Pumpkin, Inc.	many	X	co	N/A	N/A	Full	N/A	N/A
ThreadX	Express Logic, Inc.	many	X	pe	TCP/IP, BT, WLAN, IrDA, USB	FAT 16/32	X	N/A	N/A

Table 4.4: POSIX Services. POSIX Core Services is the feature set usually found in UNIX operating systems; it incorporates Standard ANSI C

POSIX.1: POSIX Core Services	
Process Creation and Control	Timers
Signals	File and Directory Operations
Floating Point Exceptions	Pipes
Segmentation Violations	C Library (Standard C)
Illegal Instructions	I/O Port Interface and Control
Bus Errors	
POSIX.1b: Real-Time Extensions	
Priority Scheduling	Message Passing
Real-Time Signals	Shared Memory
Clocks and Timers	Asynchronous and Synchronous I/O
Semaphores	Memory Locking
POSIX.1c: Threads Extensions	
Thread Creation, Control, and Cleanup	Thread Synchronization
Thread Scheduling	Signal Handling

Chapter 5

Software Architecture and Embedded Platform

5.1 Overview

The implementation task for this thesis was to decide and solve how computationally complex media codecs can be utilized in the VoIP (Voice over IP) software build on Open Multimedia Applications Platform (OMAP) processor environment. This chapter starts from requirements for the architecture and goes through the VoIP software description to Digital Signal Processor (DSP) software architecture.

The following sections define the architectural requirements, discuss the OMAP environment and its different communication schemes between the DSP and ARM (Acorn RISC Machine) processors. Finally, we take a look at the IP Telephony software and discuss the architectural decisions made for the implementation.

5.2 Requirements

The requirements described below were first defined in accordance with the implementation of Sofia-SIP [44], a Session Initiation Protocol (SIP) stack developed in Nokia Research Center in 2000-2006.

1. Complex media codecs of an IP Telephony application need to executable in real-time on the OMAP processor in Linux environment.

2. The implementation need to support simultaneous loading of multiple media codecs to DSP.

3. To utilize AMR-NB codec implementation available for the OMAP TMS320C55x

DSP.

4. Integrate the chosen AMR-NB implementation to IP Telephony software.
5. Transparent usage of DSP to media commanding system.
6. Interoperable with other AMR-NB implementations.
7. The implementation need to fulfill the algorithmic delay conditions.

5.3 Open Mobile Applications Platform

This section introduces a Texas Instruments processor architecture that is widely deployed in the field of mobile terminal processors. The major mobile communications manufacturers have reportedly decided to use Open Mobile Applications Platform (OMAP) as their mobile terminal platform [42], [77], [38].

OMAP is a dual-core architecture that encloses the both DSP and RISC processors in a single box. OMAP incorporates a TMS320C55x DSP and a TI925T ARM cores [76]. A high-level OMAP architecture is depicted in Figure 5.1. The architecture includes on-chip caches for both processors. This design is used to reduce average fetch times to external memory and to also help to eliminate the power consumption of unnecessary external accesses. The external memory (SDRAM) is accessed through Traffic Controller (TC). Each processor has separate external peripheral interfaces. This enables independent access for DMA and peripheral communication for both processors.

The periphery includes a set of timers, interrupt control, serial ports and interfaces for keyboard, camera and MMC/SD (Multimedia Card/Secure Digital), for example. Memory management units (MMUs) for both cores provide virtual-physical memory translation.

OMAP integrates two external memory interfaces and a single internal memory port. The three memory interfaces allow independent and concurrent access from either core or from the DMA unit. [40]

The OMAP architecture also provides on-chip peripherals such as timers, general-purpose I/O, a UART (Universal Asynchronous Receiver/Transmitter) and watchdog[1] timers support common OS requirements. OMAP also provides system-on-a-chip functionality with peripherals that include i.e. 192 kilobytes RAM, USB 2.0 (Universal Serial Bus), MMC/SD

[1] A watchdog timer is a hardware timer that can be used to detect software anomalies and to reset the processor if necessary. Usually a watchdog timer is based on a counter that counts down from some initial value to zero. The software selects the counter's initial value and periodically restarts it. If the counter reaches zero before the software restarts it, the software is assumed to be malfunctioning and the processor's reset signal is asserted.

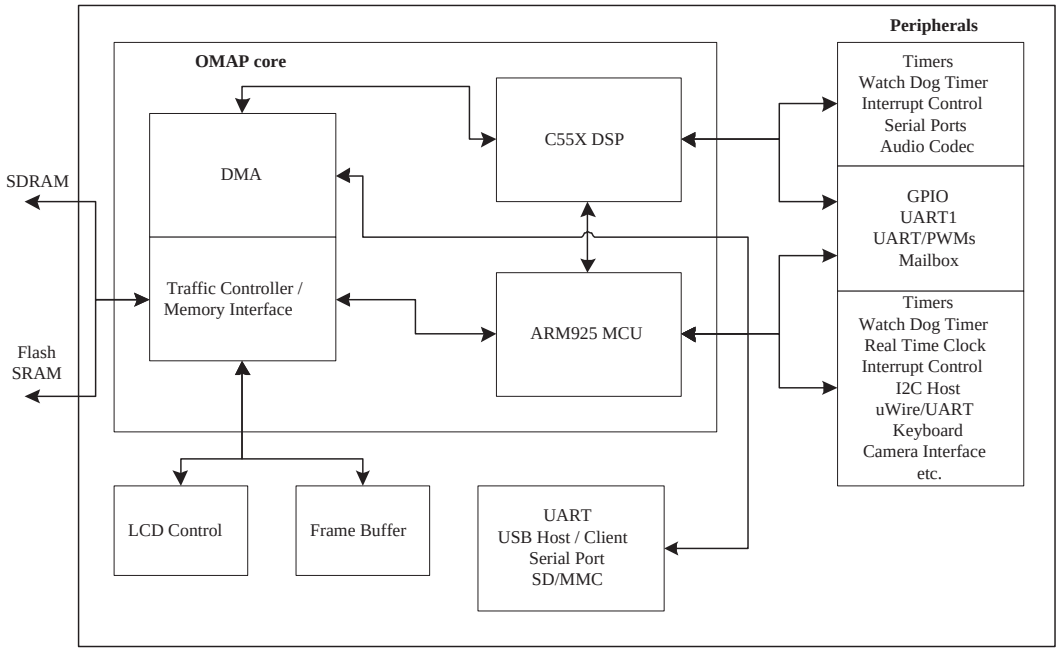

Figure 5.1: OMAP Architecture Overview. The figure depicts the OMAP core with ARM and DSP processors and the peripheral communication channels. [10]

card interface, multi-channel buffered serial ports, real-time clock, GPIO and LCD interface in addition to other features. OMAP contains a built-in interprocessor communication mechanism which provides a transparent interface to the DSP for code development.

5.3.1 TMS320C55x DSP Core

The DSP core of the OMAP device is based on the TMS320C55x DSP generation CPU processor core. The CPU supports an internal bus structure composed of one program bus, three data read buses, two data write buses, and additional buses dedicated to peripheral and DMA. These buses provide the ability to perform up to three data reads and two data writes in a single cycle. In parallel, the DMA controller can perform up to two data transfers per cycle independent of the CPU activity. [79]

The C55x CPU provides two multiply-accumulate (MAC) units, each capable of 17-bit x 17-bit multiplication in a single cycle. A central 40-bit arithmetic/logic unit (ALU) is supported by an additional 16-bit ALU. Use of the ALUs is under instruction set control, providing the ability to optimize parallel activity and power consumption. These resources are managed in the address unit (AU) and data unit (DU) of the C55x CPU. [79]

The C55x DSP generation supports a variable byte width instruction set for improved code density. The instruction unit (IU) performs 32-bit program fetches from internal or

external memory and queues instructions for the program unit (PU). The program unit decodes the instructions, directs tasks to AU and DU resources, and manages the fully protected pipeline. Predictive branching capability avoids pipeline flushes on execution of conditional instructions. The OMAP DSP core also includes a 24K-byte instruction cache to minimize external memory accesses, improving data throughput and conserving system power. [79]

5.3.2 DSP Internal Memories

OMAP1510 has three internal memories for DSP: DARAM (Dual Access RAM), SARAM (Single Access RAM) and PDROM (Program and Data ROM). DARAM and SARAM are seen in the DSP memory space and also in the MPU memory space as described in Table 5.1. The Linux memory mapping described below is specific to DSP Gateway.

Table 5.1: DSP Internal Memories and MPU Memory Mapping in Linux.

Memory section	DSP Byte Address	MPU Physical Address	Linux Virtual Address	Size
DARAM	0x000000 – 0x00ffff	0xe0000000 – 0xe000ffff	0xe0000000 – 0xe000ffff	64 kB
SARAM	0x010000 – 0x027fff	0xe0010000 – 0xe0027fff	0xe0010000 – 0xe0027fff	96 kB
PDROM	0xff8000 – 0xffffff	0xe0ff8000 – 0xe0ffffff	Not Used	32 kB

5.3.3 External Memory Mapping for DSP

DSP can also use External memory (Synchronous Dynamic RAM, SDRAM) by mapping through the DSP MMU. Using this feature, DSP application programmer can extend the available memory area for code and data which is confined in 192kB by default. This mapping procedure is handled by MPU. MPU maps the memory allocated for DSP to the DSP area in the MPU virtual space from 0xe0000000 because of following two reasons.

1. The address exchange between DSP space and MPU virtual space becomes easy when the memory is mapped to same offset in MPU and DSP.

2. If MPU accesses the shared memory (with DSP) through ordinal mapping (i.e. Linux maps area straight from 0xc0000000), synchronization problems will occur in the

MPU data cache. To avoid this case, another mapping in which the cache is disabled should be used. [30]

5.3.4 Traffic Controller

The Traffic Controller (TC) manages all accesses by the MPU, DSP, System DMA, and local bus to the OMAP memory resources (excluding the DSP internal memories, PDROM, SARAM and DARAM). The TC provides access to three different memory interfaces: External Memory Interface Slow (EMIFS), External Memory Interface Fast (EMIFF), and Internal Memory Interface (IMIF). The IMIF allows access to the 192K bytes of on-chip SRAM. [76]

The EMIFF Interface provides access to 16-bit-wide access to standard SDRAM memories and the IMIF provides access to the 192K bytes of on-chip SRAM.

The TC provides the functions of arbitrating contending accesses to the same memory interface from different initiators (MPU, DSP, System DMA, Local Bus), synchronization of accesses due to the initiators and the memory interfaces running at different clock rates, and the buffering of data allowing burst access for more efficient multiplexing of transfers from multiple initiators to the memory interfaces. [76]

The architecture of TC allows simultaneous transfers between initiators and different memory interfaces without penalty. For instance, if the MPU is accessing the EMIFF at the same time, the DSP is accessing the IMIF, transfers may occur simultaneously since there is no contention for resources. There are three separate ports to the TC from the System DMA (one for each of the memory interfaces), allowing for greater bandwidth capability between the System DMA and the TC. [76]

5.4 Interprocessor Communication of OMAP

Communication between the ARM and DSP processors is enabled by a mailbox communication scheme. Communication is achieved when one processor writes to the appropriate command word register which causes an interrupt to the other processor and sets the appropriate flag register

The interrupted processor acknowledges by reading the command word which causes the flag register to be cleared. An additional data-word register is also available in each mailbox register set to optionally communicate two words of data between the processors for each interrupt instead of just communicating the command word

There are three sets of mailbox registers. The first one is for MPU to send messages and issue an interrupt (INT5) to DSP. The other two are for DSP to send messages and issue interrupts (IRQ10/IRQ11) to MPU. Each set of mailbox registers consists of two 16-bit

registers and a 1-bit flag register. The interrupting processor can use one 16-bit register to pass data word to the interrupted processor and the other 16-bit register to pass a command word. The protocol used in DSP Gateway is defined here. [30]

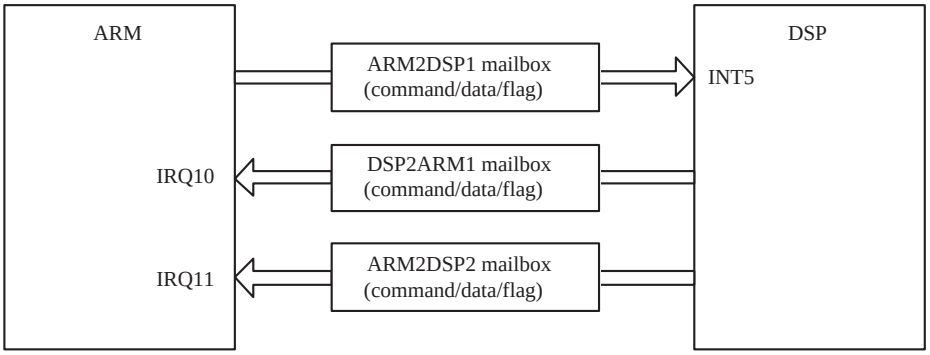

Figure 5.2: OMAP Mailbox Communication Scheme. ARM communicates through ARM2DSP1 mailbox. DSP is able to communicate through both DSP2ARM mailboxes.

5.4.1 Shared Memory Utilization

Shared memory architecture is implemented via the Traffic Controller. It can be used in conjunction with the mailbox registers handshaking interrupts for synchronizing the MPU and DSP accesses to shared memory.

Utilizing the shared memory is useful when the data to be passed between the MPU and DSP is larger than the two 16-bit words of mailbox command and data registers. For example, the MPU may need to provide the DSP with a list of pointers to perform a specific task as opposed to a single command and single pointer.

DSP could read the list of pointers from shared memory after receiving the interrupt caused by an MPU write to the mailbox command register.

5.4.2 MPU Interface

MPU Interface (MPUI) allows the MPU and the system DMA controller to communicate with the DSP and its peripherals. MPU allows access to the full memory space (16M bytes) of the DSP and the DSP public peripheral bus.

Both MPU and DMA Controller have the read and write access to the complete DSP I/O space (128K bytes), which includes control registers of the DSP public peripherals. DSP I/O space can be used for many functions, for example for MPU loading of program code into DSP program memory space, for data sharing between MPU and DSP interprocessor communication protocols via shared memory, and for MPU to use and control DSP public

peripheral buses.

5.4.3 DSP Gateway

DSP Gateway is a piece of software that enables data transmission on the OMAP between the MPU processor running Linux and the DSP running DSP/BIOS II operating system.

The DSP Gateway consists of two parts, a Linux device driver on the ARM-side and a library on the DSP-side which communicate with each other through the mailbox. The driver provides conventional Linux device driver interface so the application programmers can use the DSP through system calls such as `read()`, `write()` and `ioctl()`. The library on DSP provides interface for DSP tasks which are utilized from Linux. The list below describes the main characteristics of the DSP Gateway functionality. [30]

- Only one of two mailboxes from DSP to ARM is used (DSP2ARM1).
- Tasks run in DSP are identified by Task ID (TID).
- Data transferred through mailboxes is multiplexed. TID is enclosed in mailbox commands to distinguish task data from each other.
- Interprocessor buffers (IPBUFs) for the block data transfer between ARM and DSP are defined. These buffers can be seen by both processors. The Global IPBUFs are identified with IDs (BID).
- IPBUFs have ownerships. Only the processor which has the ownership of the IPBUF can use that IPBUF. When a data transfer with an IPBUF is performed between the processors, the ownership of the IPBUF is passed on to the recipient.

5.5 Sofia-SIP Internet Telephony Software Suite

The implementation code described in this thesis was integrated to a IP Telephony application, Sofia-SIP[2] [44], a Session Initiation Protocol (SIP) stack developed in Nokia Research Center. Figure 5.3 depicts the software architecture in hierarchical modules. In general, the software implements the signaling and media components needed in establishing a variety of IP Telephony and media streaming sessions. The signaling module integrates IETF specifications for SIP 2.0, HTTP 1.1 and RTSP 1.0 protocols and numerous related RFCs and Internet Drafts.

The Applications layer provides several applications from light weight embedded terminal software to a number of heavily loadable network elements including SIP, RTSP and HTTP related services, each of them utilizing the lower level modules.

[2]Signalling part of Sofia-SIP was released as Open Source in July 2005 and the media part is based on a different system than described in this thesis.

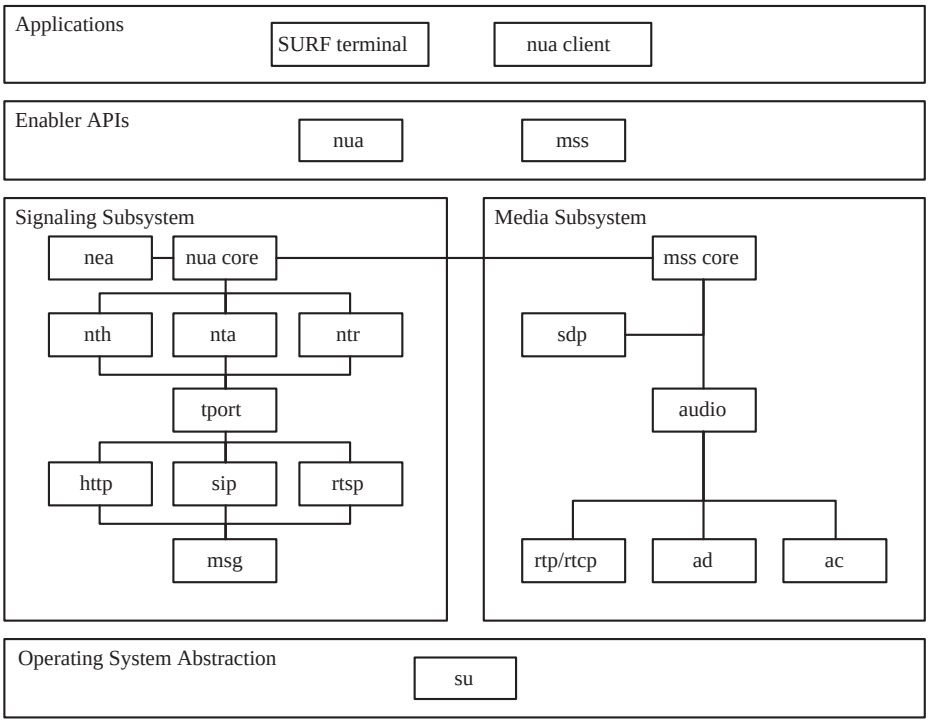

Figure 5.3: Sofia-SIP Internet Telephony Software Suite. The figure presents the current Sofia-SIP architecture with hierarchical drawings of software module dependencies.

The Enabler API offers interfaces for signaling and media handling for Applications layer. The `nua` implements functionality for creating signaling services for protocols such as SIP, RTSP and HTTP. The `mss` refers to Media Subsystem (MSS) and provides interfaces for media control. A short description of each module is presented below, more in-depth documentation of the modules is presented in Mobile Internet Technical Architecture (MITA) book series [41].

- `SURF terminal`: IP Telephony communications software with graphical user interface
- `nua`: Signaling subsystem API for Applications layer
- `nea`: Presence event enabler
- `nua` core: The signaling transaction handler
- `nua client`: a command line (text based) IP Telephony communications software
- `mss`: Media Subsystem API for media control
- `nta`: SIP state machine
- `ntr`: RTSP state machine
- `nth`: HTTP state machine

- `tport`: Signaling transport abstraction
- `http`: HTTP message parser
- `sip`: SIP message parser
- `rtsp`: RTSP message parser
- `msg`: Signaling message serializer

- `mss` core: Lower level media control services
- `sdp`: Session Description Protocol (SDP) parser
- `audio`: Audio control interface
- `rtp/rtcp`: RTP/RTCP control module
- `ad`: Media device abstraction interface
- `ac`: Media codec interface

- `su`: Network and memory service library with additional support utilities

5.6 DSP Software

The DSP Gateway was chosen for interprocessor communication it being the only Linux implementation available. The developers of the DSP Gateway provided fast and extensive support for building applications on top of the gateway.

Figure 5.4 depicts a high-level abstraction of the DSP-enabled SURF architecture. The ARM-side is divided in User space and Kernel space. User space serves as a playground for conventional applications. Kernel space includes drivers and kernel modules, for example.

The DSP Gateway is in the in the bottom of the figure. The interprocessor communication is done through the message exchange system called as mailbox. The data transfer is executed through the common memory interface. DSP Gateway provides the functionality for the communication schema and task control. Each task is communicating through the Gateway.

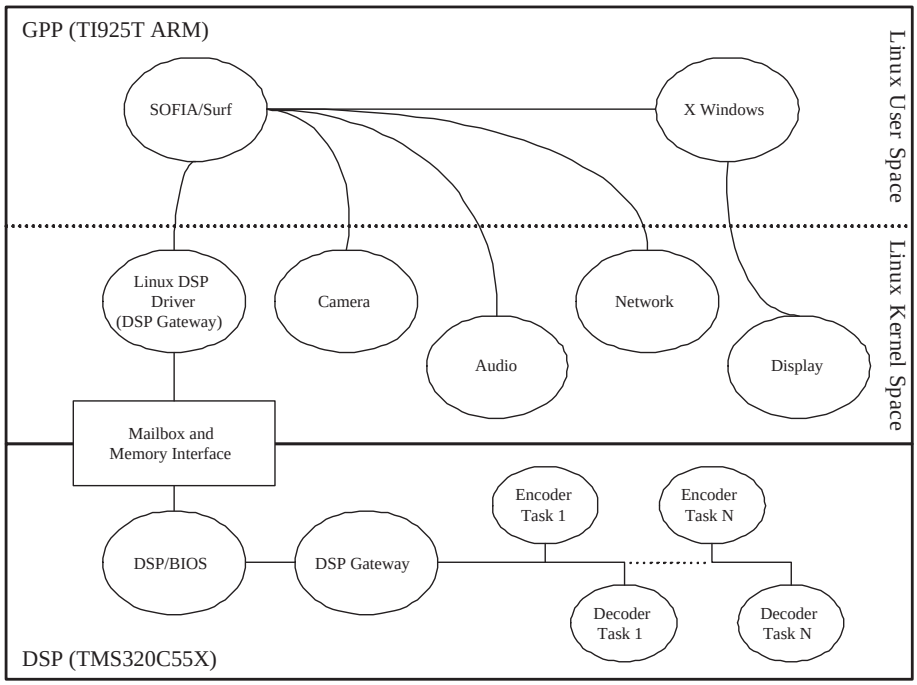

Figure 5.4: High-level Abstraction of Sofia-SIP with DSP Gateway.

Chapter 6

Implementation

6.1 Overview

The codec (coder/decoder) to be integrated for this thesis was chosen to be Adaptive Multi-Rate (AMR) speech codec. In the first measurements it was clearly shown that the general purpose processor (GPP, MPU) of OMAP (Open Multimedia Applications Platform) was not efficient enough to execute this codec in realtime. The 20 millisecond algorithmic upper boundary for encode-decode process for one speech frame was exceeded even no other processes were executed on the test platform. The logical solution for integration was to utilize the DSP processor of OMAP. The results are presented in Chapter 7.

The target of the implementation work was to integrate a DSP version of AMR speech codec to a Sofia-SIP IP Telephony application running on Linux in ARM925T GPP [1] of OMAP dual-core processor.

6.2 The Code Development Environment

The hardware used was the evaluation module of OMAP1510 DC EVM (later "OMAP EVM"), ARM-side was configured for 120 MHz and DSP-side for 120 MHz, too. According to specifications, the maximum DSP clock frequency is 160 MHz but this was never tested because the hardware was expensive and great-results-by-over-clocking was not the approach in consideration.

To ensure a closed and secure testing environment OMAP EVM was connected to a local test network with a Linux PC Gateway (Figure 6.1). The latter was also connected to local intranet and software binary directory was NFS (Network File System) mounted from a

[1] The ARM925T processor of OMAP is referred here as General Purpose Processor (GPP) to distinguish it from OMAP DSP core, TMS320C55.

Linux file server. EVM was switched on and Linux package was loaded from a Windows PC to EVM by the parallel port connected Lauterbach debugger.

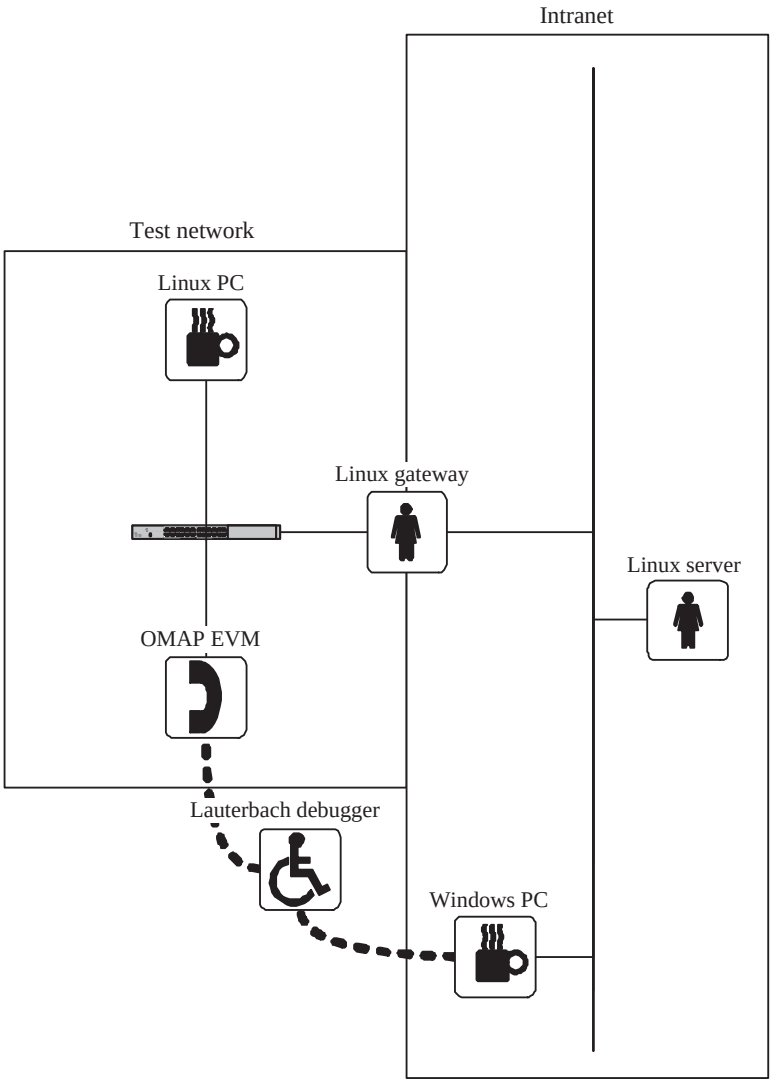

Figure 6.1: Test Network Setup for OMAP EVM.

OMAP EVM was connected to Ethernet and to a debugging hardware (Lauterbach) that was used to load a minimal Linux distribution to EVM RAM. A special setup was build to enable easy loading of DSP software. Figure 6.2 depicts the phases of setting up the OMAP EVM environment.

After successfully booting OMAP EVM a loading script was fetched by WGET (FTP, File Transfer Protocol) from PC Gateway. The shell script loaded every component of the software under development, including the DSP software module, speech samples, DSP

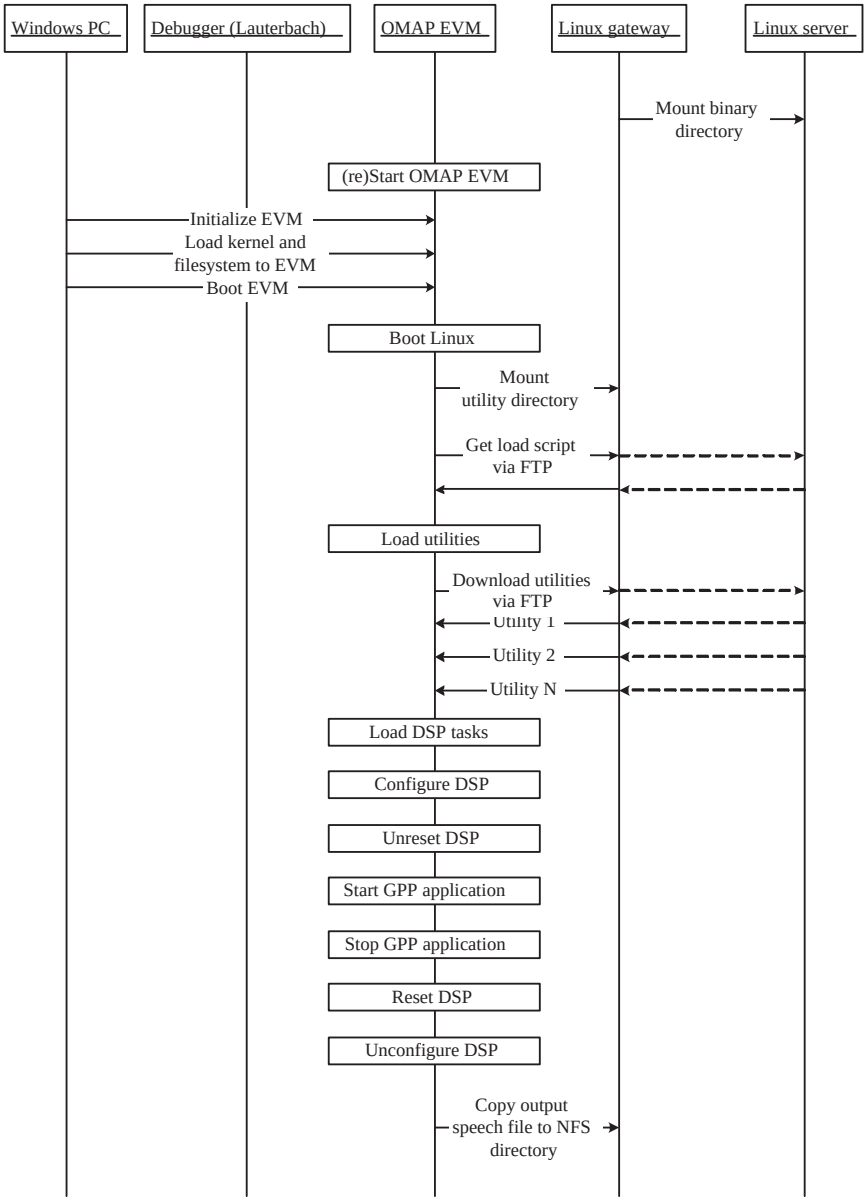

Figure 6.2: Test Environment Setup for OMAP EVM.

configuration utility `dspctl` and several test applications.

`dspctl` was used to map external memory, SDRAM, for DSP and to load the DSP binary from Linux to DSP. When DSP codec tasks were successfully initialized the test application or the actual IP Telephony software was launched in Linux. The application connected from Linux to DSP codecs.

6.2.1 Software

The DSP specific code development environment used was Code Composer Studio by Texas Instruments running on a Windows machine. The software provided necessary libraries and configuration tools for building DSP applications for Texas Instruments' OMAP and more precisely TMS320C5XX processor platforms. Instead of using Texas Instruments DSP/BIOS Bridge an Open Source version of DSP Gateway [43] was used. DSP-side code was linked to DSP Gateway library (TOKliBIOS), the DSP operating system library that provided the task control schema between DSP and Linux DSP kernel module.

The application processor was running Linux kernel version 2.4.19 with ARM Linux, OMAP and DSP Gateway patches. The Linux-side application code was compiled with GNU C Compiler (GCC).

6.3 Integration

The integration work consisted of attaching the MPU-side codec abstraction interface to DSP-side codec instances. The connecting blocks for both sides were implemented. The integration on MPU-side was nevertheless trivial due to good design of existing codec APIs and simple MPU-side use model of DSP Gateway. Figure 6.3 depicts the communication scheme between the AC module of the Sofia-SIP application and the DSP Gateway interfaces.

The challenging part of the implementation was the utilization of DSP-side of OMAP processor. The first versions of DSP Gateway were lacking the functionality for sending debug messages from DSP to MPU-side. The evaluation of DSP functionality was nearly black-box oriented. The content of a DSP-MPU data transmission buffer was examined subjectively to fix erroneous functionality.

Segmentation fault on DSP-side caused the Linux to malfunction and the restarting and reloading the packages to Linux' RAM took several minutes. The troubleshooting was dramatically accelerated after the release of DSP Gateway version 2.0. The new version provided enhanced stability and debugging capabilities, including printing debug messages through DSP Gateway to Linux console.

The file listing below shows four DSP tasks created by running `dspctl` utility for final DSP media codec application. The first two file I/O devices are the AMR decoder and encoder tasks, respectively. The following two devices are the G.711a codec tasks.

```
crw-rw-rw-    1 0    0    97,   3 Jan 1 00:00 /dev/dsptask/amr_dec
crw-rw-rw-    1 0    0    97,   2 Jan 1 00:00 /dev/dsptask/amr_enc
crw-rw-rw-    1 0    0    97,   1 Jan 1 00:00 /dev/dsptask/g711a_dec
crw-rw-rw-    1 0    0    97,   0 Jan 1 00:00 /dev/dsptask/g711a_enc
```

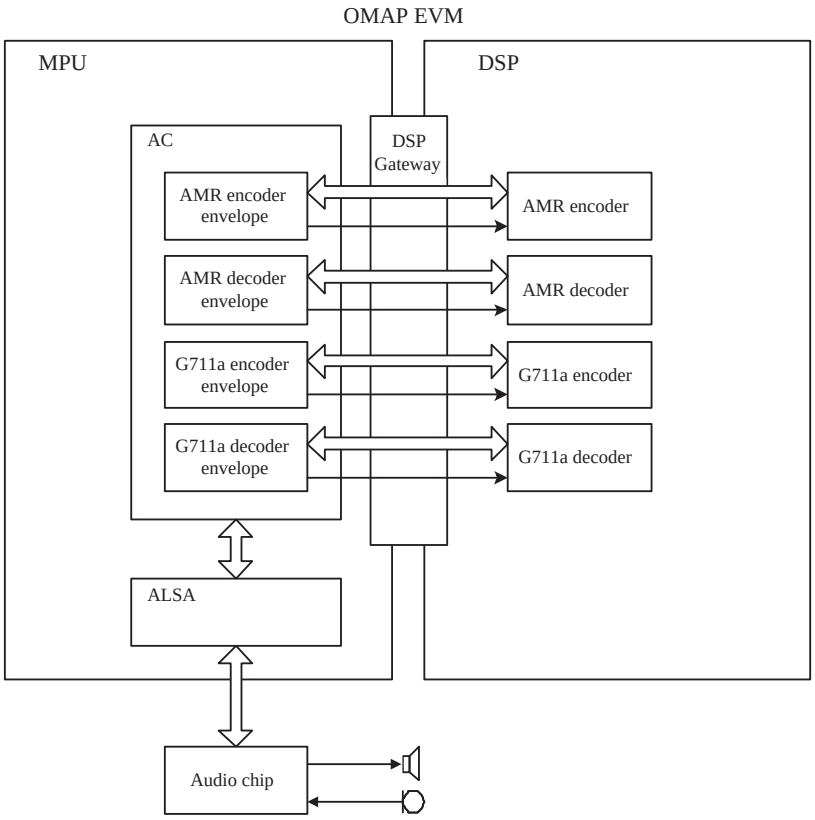

Figure 6.3: Architectural Description of Sofia-SIP IP Telephony Suite's Media Subsystem Integration with DSP Gateway and DSP Media Codecs. The wide arrows represent channels where the speech data is transmitted and received. Black arrows represent the control channels.

6.3.1 DSP Memory Utilization

As described in Sections 5.3.2 and 5.3.3, OMAP is equipped with several different physical memory segments. The variables and codec specific tables were located carefully to lower the delay caused by additional memory I/O operations. For example, the data transfer buffer between Linux kernel and a DSP task was placed in Dual Access RAM (DARAM) to allow memory transfer without interrupt of Traffic Controller (TC).

Chapter 7

Measurements and Results

7.1 Measurements

The first trials with the Sofia-SIP IP Telephony application's media subsystem were conducted with the setup depicted in Figure 7.1. The test application was `AC_TEST` (Audio Codec Test Application). An example speech file coded as 16-bit linear PCM (Pulse Code Modulation) was read in frames of 160 samples (320 bytes) by the measurement applications.

Each frame was written to AMR (Adaptive Multi-Rate) encoder DSP task followed by a subsequent read of 118 bytes. The encoded payload was passed to RTP (Real-Time Transport Protocol) packetization module. The output refers to RTP payload format for AMR. The RTP packet was passed to RTP depacketization module and subsequently passed to DSP by writing to AMR decoder DSP task. The encoded AMR frame was decoded by the DSP task and finally read by the media subsystem.

The resulting 16-bit coded 320 byte frames were subsequently written to a temporary file. Finally, the PCM file was converted to the WAV (Waveform Audio Format) format. Several subjective listening tests were performed with loudspeakers and a headset. Perceptual quality of the processed (encoded and decoded) speech resembled the original recording.

The first test with the SURF terminal application was performed over the network as depicted in Figure 7.2. Testing was done over the Ethernet with Linux laptop on the other end. The both OMAP EVM and the laptop were executing the same version of SURF terminal, except OMAP EVM was using the command line version of the user interface and was armed with the additional DSP capability. The ARM codec in Linux laptop was based on the 3GPP reference C code with the additional proprietary RTP payload format implementation.

The setup was similar to Figure 7.1 with the additional network and software loopback

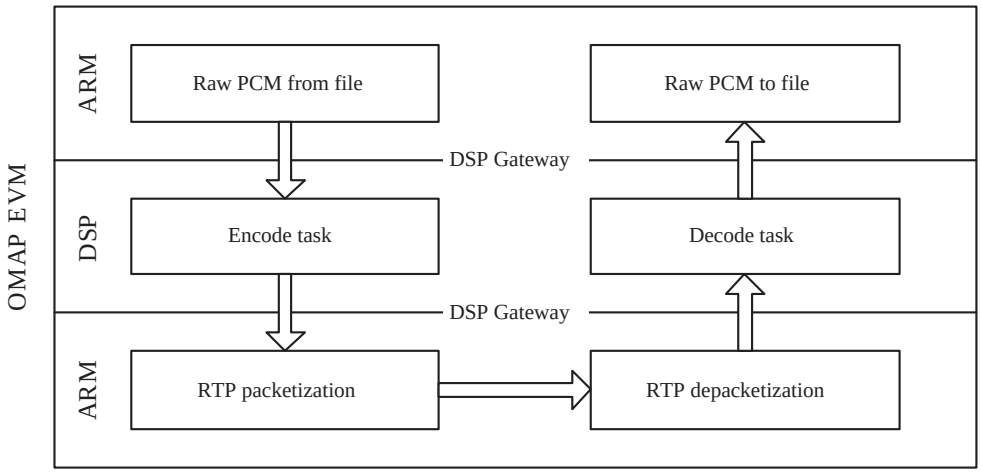

Figure 7.1: Testing Loop for Encode-Decode Process.

provided by Media Subsystem. To enable reproducible measurement scheme a pre-recorded speech file was used as an input for Surf on a Linux laptop. This data was transmitted in real-time to OMAP EVM and rerouted back using built-in software loopback of Media Subsystem.

`nua_cli` was modified to contain a measurement scheme that was utilized to measure delay of different phases of AMR coding. A timestamp was stored before and after executing encode, decode, RTP payload packetization and RTP payload depacketization. The difference of the timestamps around a function is the delay or latency one operation creates.

7.2 Results

Figure 7.3 depicts the delay measurements of 3GPP reference implementation of AMR codec that was previously integrated to Sofia-SIP. The codec was executed on General Purpose Processor (GPP), and in OMAP this is the ARM-side. It can be clearly seen that the average of the encoding operations lasts 16 ms and decoding operations 7 ms. Real-time requirements can not be met if both encoder and decoder are executed simultaneously due to 20 ms speech frames and algorithmic delay ($16ms + 7ms = 23ms$, see Figure 7.3). This result was also verified by setting up an RTP session (using SIP) between the both ends as described in Figure 7.2.

The codec specific parameters for below mentioned tests were chosen to maximize the load caused by the codec. Voice Activity Detection (VAD) was disabled to enable constant algorithm execution, and the bit rate was set to maximum of 12.2 kbps. Figure 7.4 depicts the results of using DSP version of AMR with DSP Gateway on OMAP. The GPP-side

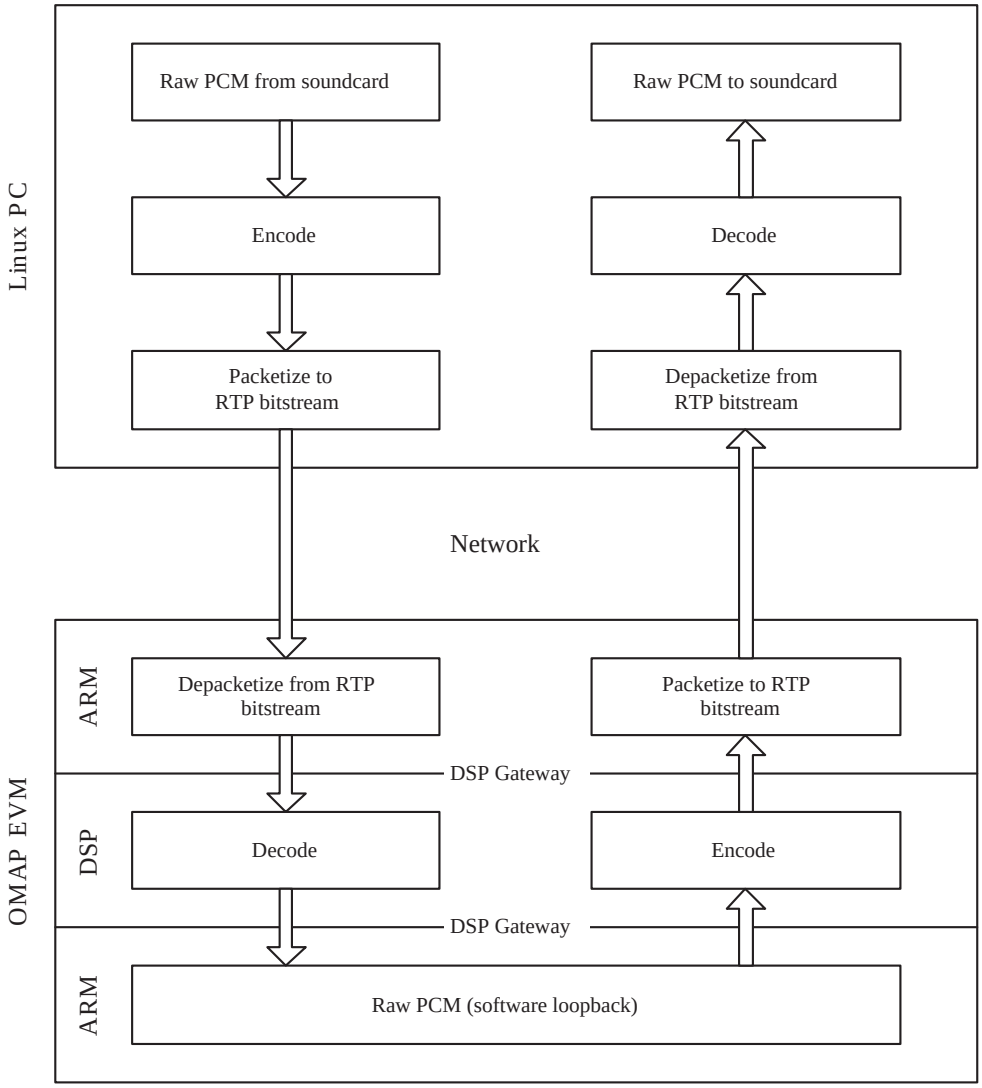

Figure 7.2: Interoperability Testing Loop for Encode-Decode Process.

application `AC_TEST` provided synchronous[1] encode/decode operations.

The measurements were made after the code and data sections of DSP binary were carefully relocated to optimal memories. Code sections of DSP code were placed in external memory, SDRAM. Data sections (including AMR codec tables) were located in SARAM. The DSP Gateway mailbox buffer, `IPBUF` was placed in DARAM to enable fast access from both GPP and DSP tasks. This optimization is referred as "Source Code Optimization" in the captions of the following pictures.

[1] Also called as blocking. Execution of the process halts until it has been woken up by the operating system. In this case synchronous refers to waiting answer from the DSP by using `read()` function call.

Figure 7.5 depicts the results of using similar setup as in Figure 7.4 with the difference of changing the GPP-side process scheduling to `SCHED_FIFO`. This set the process of `AC_TEST` to the beginning of kernel scheduler event loop. This is clearly seen as decrease of coding latency variation, and the results are more deterministic in terms of coding duration. This can be explained by the process priorization: the process of `AC_TEST` was executed first every time it was ready to continue after the sleep period of a blocking `read()` or `write()` system call.

Figure 7.6 depicts the results of using DSP version of AMR when the memory sections were not optimally chosen. The encoding delay varies in three levels with the average of 5 ms.

Figure 7.7 shows the dramatical decrease in the delay after the memory sections were located as explained above. The encoding delay is around 3.7 ms and a decoding operation is performed in average of 1.5 ms. The ripple of encoding and decoding close to 10 ms was deduced to be caused by the experimental network (Ethernet) driver. The network driver caused kernel to stall for a certain period of time. This conclusion is confirmed by the results of Figures 7.4 and 7.5 where network was not utilized.

Figure 7.8 depicts the results of using `SCHED_FIFO` priorization with the configuration similar to aforementioned. Some ripple can still be noticed around 10 ms. In this respect the changes in priorization did not affect in codec performance.

Interestingly, the decoding operations are more exposed to ripple than encoding. This can be explained as follows. Network driver is receiving packets around every 20 ms due to the algorithmic delay of AMR-NB coding scheme. Due to software loopback of IP Telephony software `nua_cli` in OMAP, RTP depacketization and decoding operating is executed first and immediately followed by the encoding and RTP packetization. After this the packet is assigned to the network driver. At this point the network driver can possibly be transmitting the packet and receiving a new one. When this happens at the same time as the next decoding operation is blocking – `write()` or `read()` issued to kernel and DSP Gateway – the process of `nua_cli` is sleeping until the kernel wakes it.

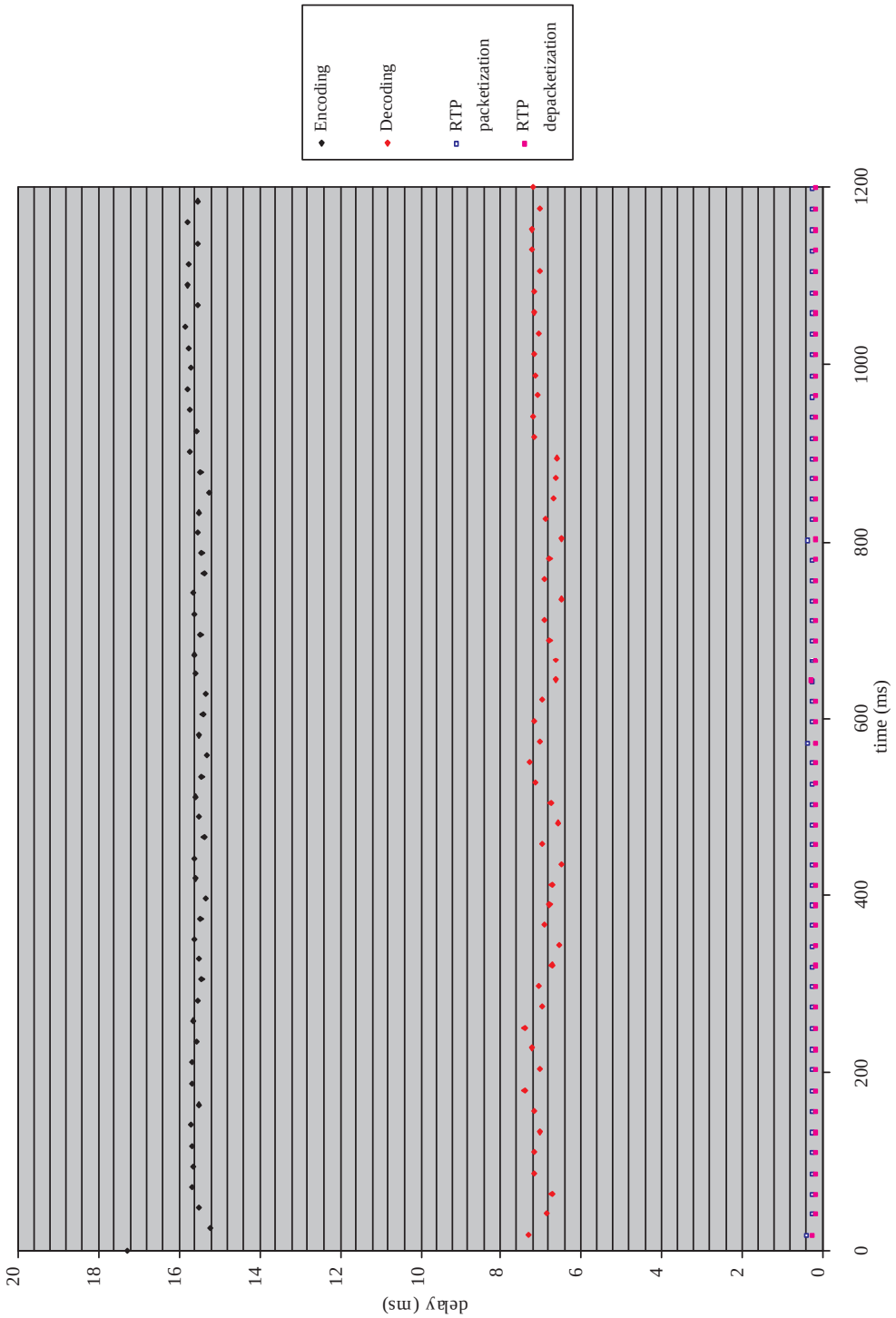

Figure 7.3: Delay Measurements for ARM-side AMR with AC_TEST without Scheduling Priorization.

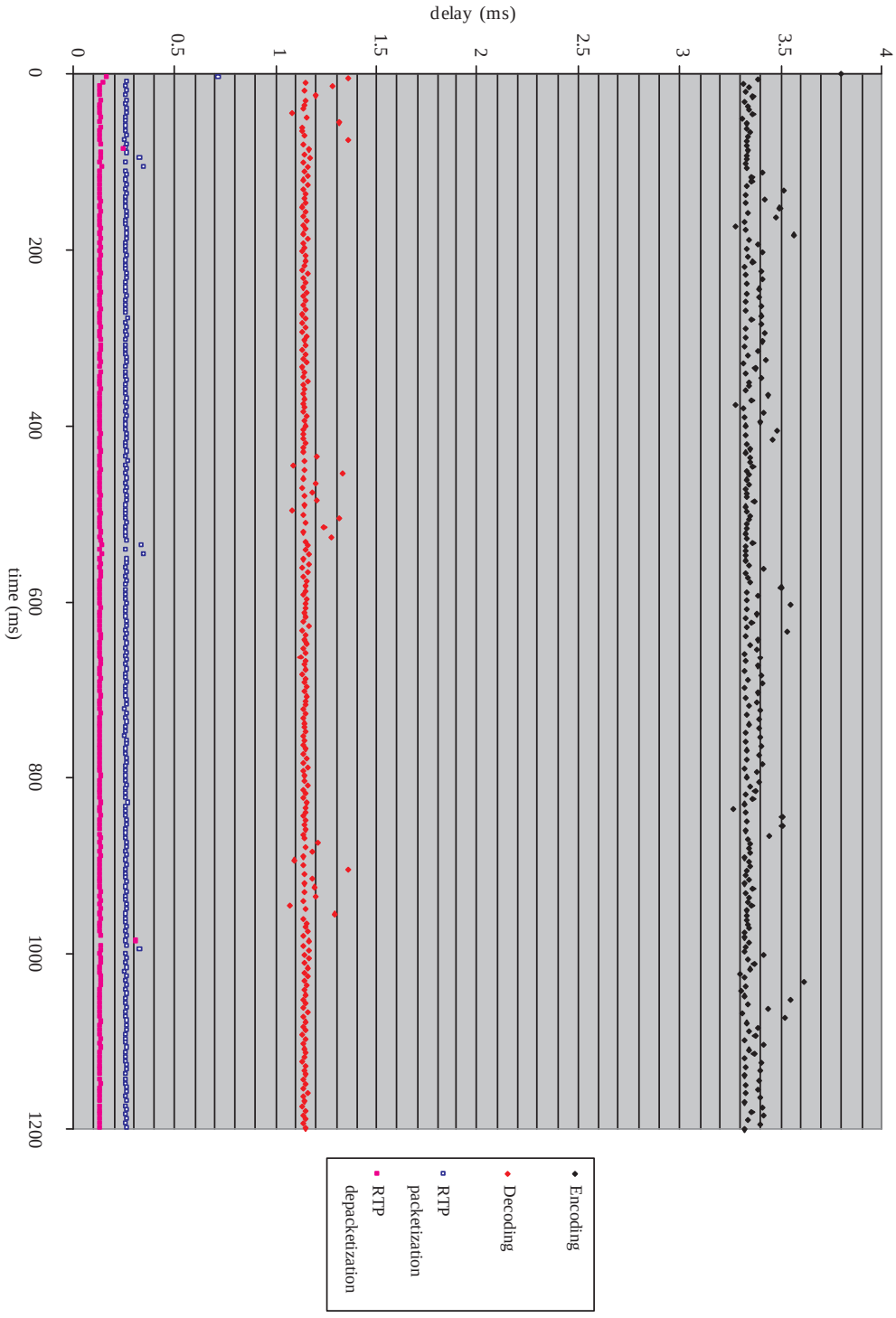

Figure 7.4: Delay Measurements for Source Code Optimized AMR with AC_TEST without Scheduling Priorization.

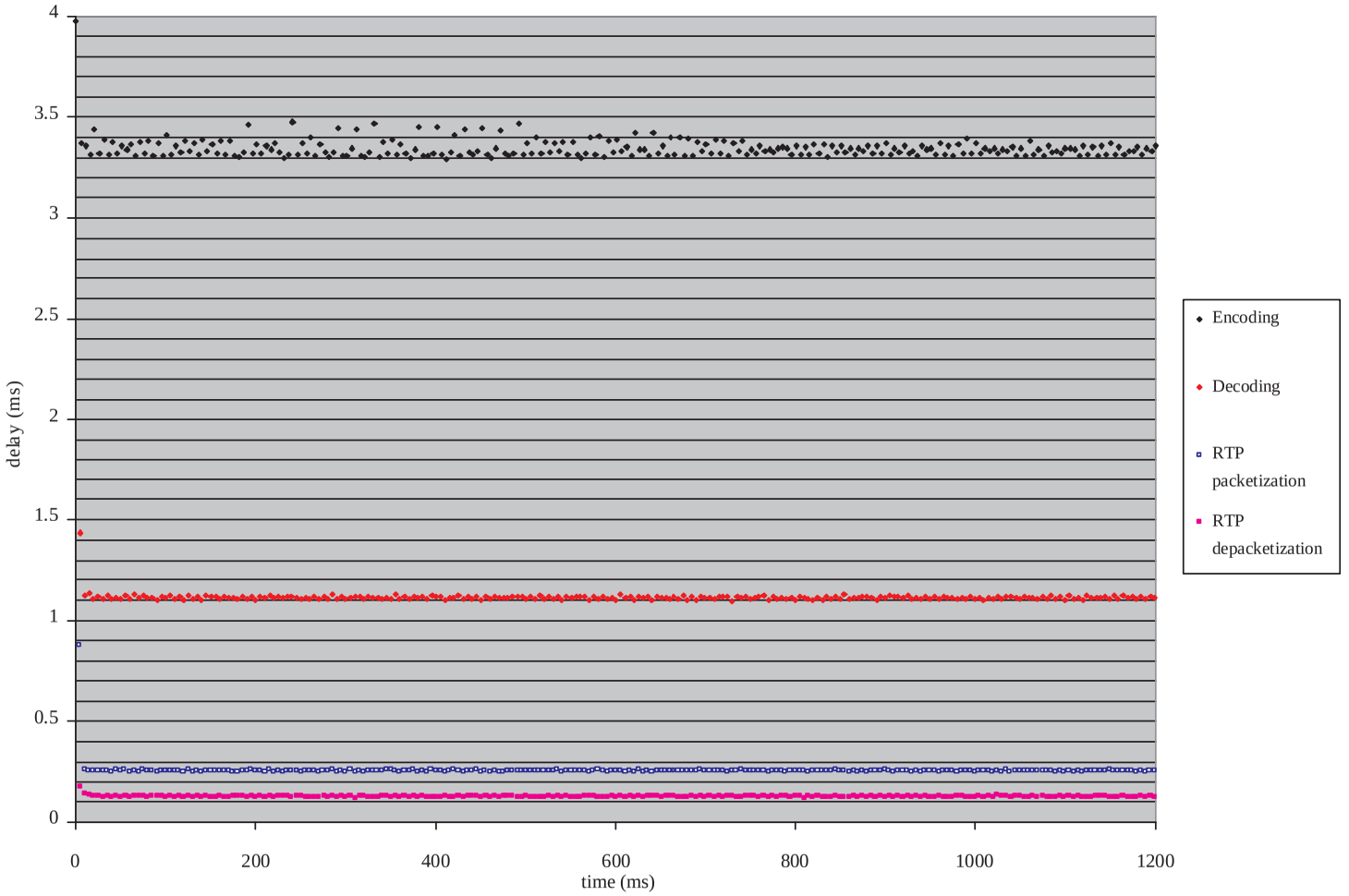

Figure 7.5: Delay Measurements for Source Code Optimized AMR with AC_TEST with Scheduling Priorization.

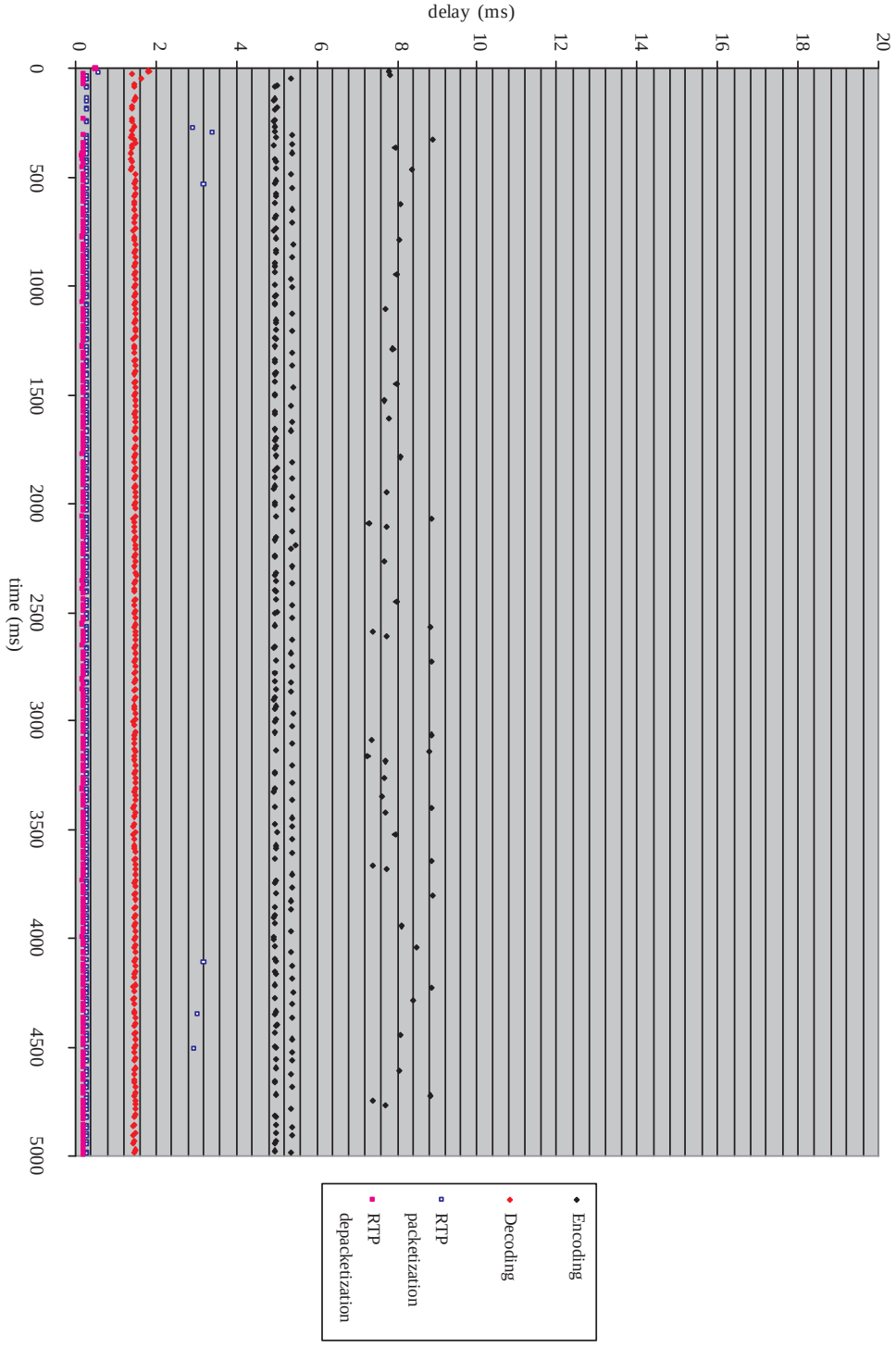

Figure 7.6: Delay Measurements for Unoptimized AMR with NUA_CLI without Scheduling Priorization.

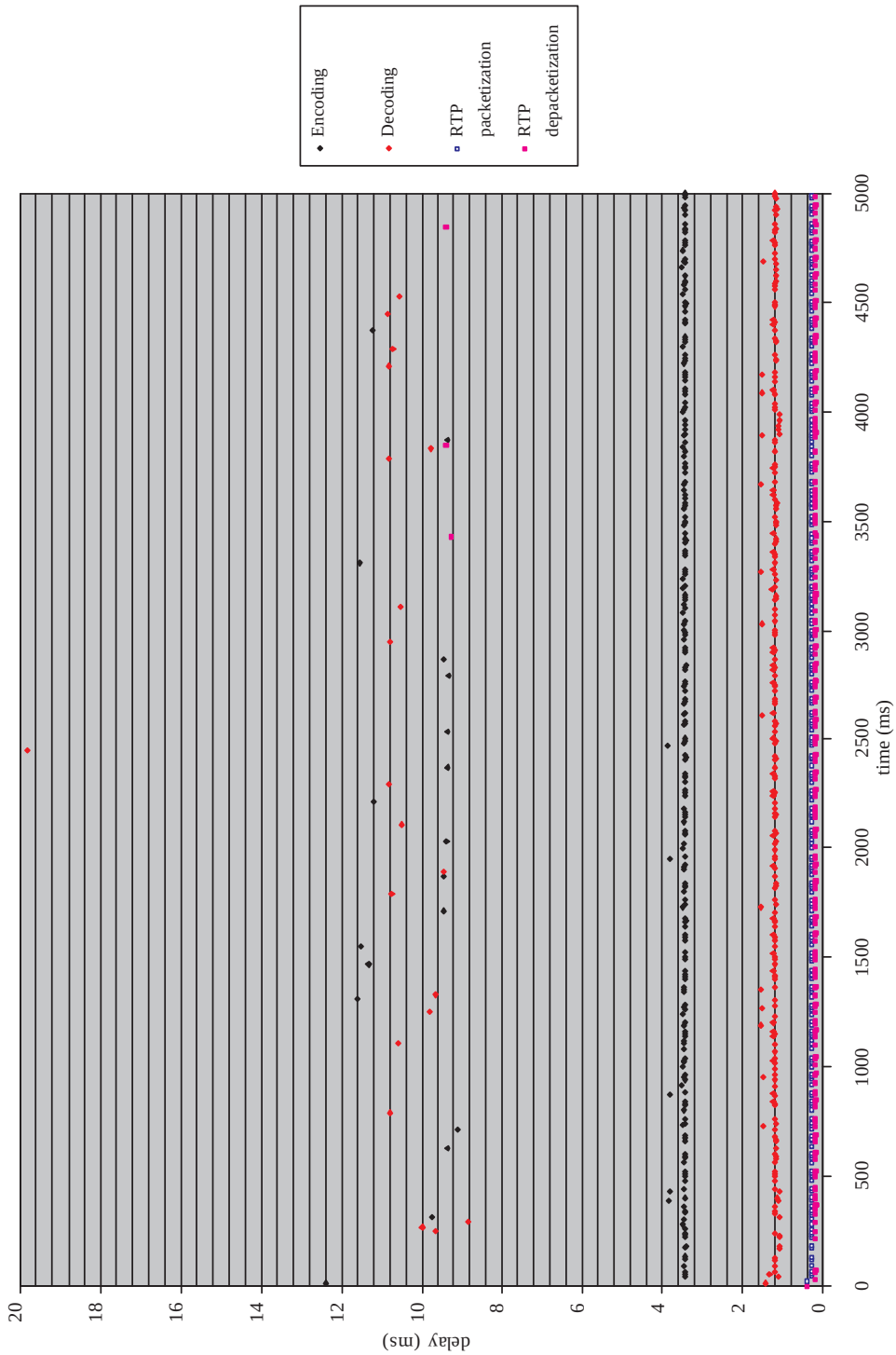

Figure 7.7: Delay Measurements for Source Code Optimized AMR with NUA_CLI without Scheduling Priorization.

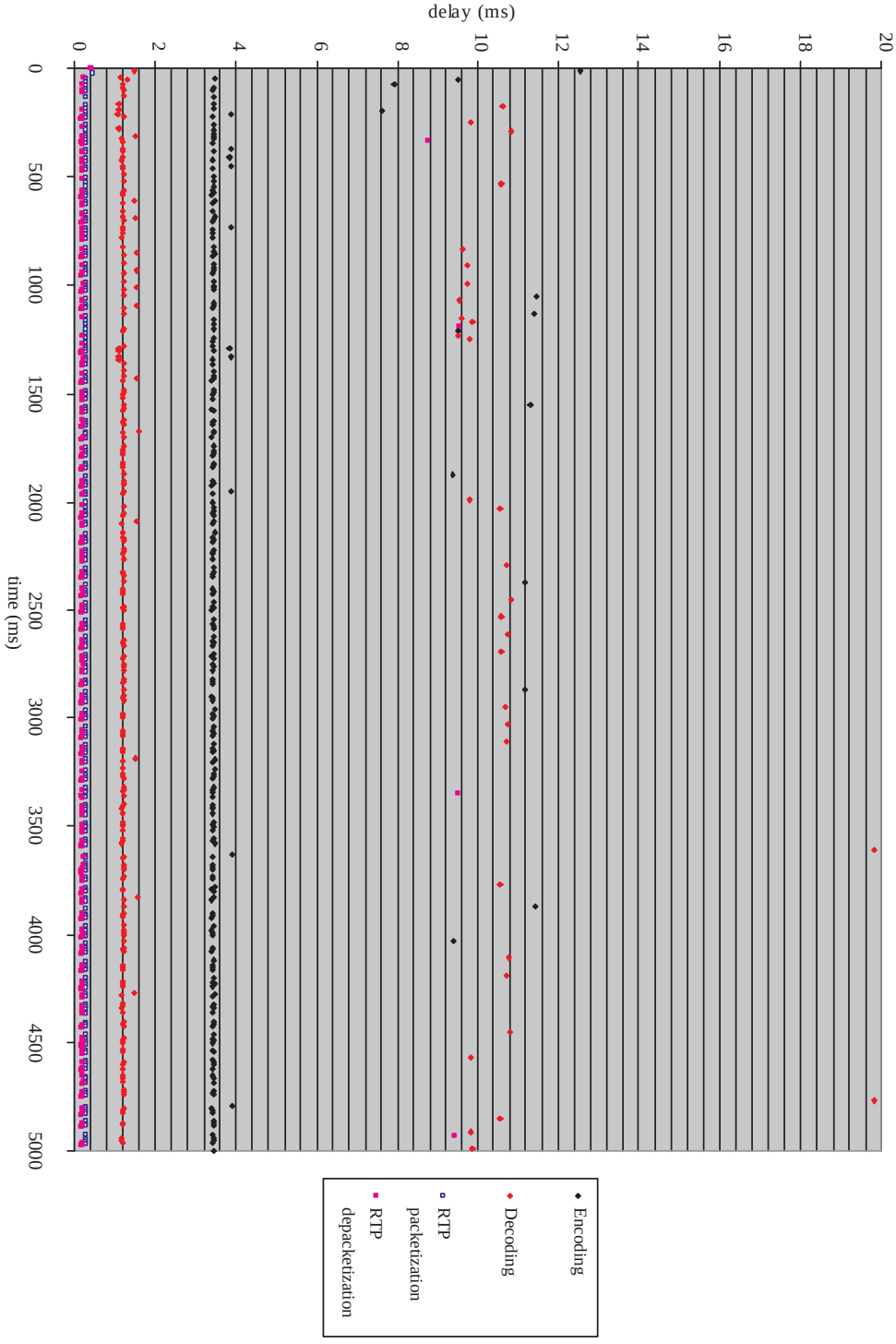

Figure 7.8: Delay Measurements for Source Code Optimized AMR with NUA_CLI with Scheduling Priorization.

Chapter 8

Conclusions

This thesis focused on speech coding in mobile terminals attached to IP networks. The background study contained media streaming and communication protocols, speech coding schemes in mobile environment and different DSP architectures.

Requirements were set for the joint software architecture of the IP Telephony application and the embedded mobile platform. The architecture was chosen and designed based on the requirements, and the implementation work was successfully carried out based on the gained knowledge. The DSP version of the AMR-NB (Adaptive Multi-Rate Narrow Band) speech codec (coder/decoder) was integrated to Sofia-SIP Internet Telephony application and the goal set in Introduction (Chapter 1) was clearly reached. The estimated time frame for implementation was slightly exceeded due to demanding task of DSP-side programming without proper documentation.

Comprehensive measurements were conducted in order to provide reliable results of the implementation work done. Results show the difference in performance with and without DSP and with several parameters. It is possible to achieve real-time functionality by using the implemented architecture.

Current mobile devices have microprocessors with a limited performance to execute demanding real-time tasks like full-screen video and good quality audio simultaneously with other tasks. High-level interoperability between the application processor and the signal processor makes it possible to take full advantage of the limited resources. The architecture implemented in this thesis serves as a good and pioneering example of how to integrate computationally complex media algorithms into an embedded device with a dual-core processor.

References

[1] 3GPP. *IP Multimedia Subsystem (IMS); Stage 2 (Release 5)*. 3GPP, 2002.

[2] 3GPP. *TS 26.071: Adaptive Multi-Rate (AMR) speech codec; General Description*, June 2002.

[3] 3GPP. *TS 26.090: Adaptive Multi-Rate (AMR) speech codec; Transcoding Functions*, June 2002.

[4] 3GPP. *Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 6)*. 3GPP, 2003.

[5] 3rd Generation Partnership Project (3GPP). 3GPP home page, 2003. http://www.3gpp.org/.

[6] Andrew Bateman and Iain Paterson-Stephens. *The DSP Handbook: Algorithms, Applications and Design Techniques*. Prentice Hall, 2002.

[7] Cisco Systems, Inc. *Architecture for Voice, Video and Integrated Data*, 2000. http://www.oneunified.net/pdf/Architecture.pdf.

[8] Joseph Cole, Ronald A.and Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue. *Survey of the State of the Art in Human Language Technology*. National Science Foundation, European Commission, 1995. http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html.

[9] J.D. Day and H. Zimmermann. The OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on*, 28:425–432, April 1980.

[10] Sebastein De-Gregorio, Madhukar Budagavi, and Jamil Chaoui. *Bringing Streaming Video to Wireless Handheld Devices*. Texas Instruments, Inc., May 2002. White Paper.

[11] Express Logic. ThreadX, 2003. http://www.expresslogic.com/.

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[13] Free Software Foundation. The GNU C Library, 2001. http://www.gnu.org/.

[14] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, November 1996.

[15] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046, November 1996.

[16] Giovino, B. Microcontrollers and DSPs – Will They Converge? *Embedded System Programming*, September 2001. http://microcontroller.com/wp/MicrosDSPs/micros_dsps.htm.

[17] L. Godell, M. M. Nordan, and T. Lapolla. The Price of VOIP in Europe. *Forrester Research; TechStrategy Report*, December 2002.

[18] Google. GoogleTalk, 2006. http://www.google.com/talk/.

[19] GSM Association. GSM World - GPRS Platform, 2003. http://www.gsmworld.com/technology/gprs/index.shtml.

[20] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, Internet Engineering Task Force, April 1998.

[21] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. Internet draft, Internet Engineering Task Force, June 2003.

[22] HiperLAN2. HiperLAN2 Global Forum, 2003. http://www.hiperlan2.com/.

[23] IEEE. *The Open Group Technical Standard Base Specifications, Issue 6. Std 1003.1*, 2003.

[24] ITU-T. *"Recommendation G.711 Pulse Code Modulation (PCM) of Voice Frequencies*, 1988.

[25] ITU-T. *"Recommendation H.323 Packet-based Multimedia Communications Systems*, 2000.

[26] ITU-T. *"Recommendation P.862 Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs* , 2001.

[27] ITU-T. *Study Group 16, Question E; Media Coding Summary Table*, May 2003.

[28] Jax, P. and Vary, P. Wideband Extension of Telephone Speech Using a Hidden Markov Model. *IEEE Workshop on Speech Coding*, September 2000.

[29] JMI Software Systems, Inc. C-Executive and PSX, 2004. http://www.jmi.com/cexec.html.

[30] Kobayashi, Toshihiro. *Linux DSP Gateway Specification, Revision 2.0*. Nokia, Inc., November 2003. http://sourceforge.net/projects/dspgateway/.

[31] KROS Technologies. KROS Operating System, 2004. http://www.krostech.com/.

[32] LynuxWorks, Inc. What is POSIX?, 2003. http://www.lynuxworks.com/products/posix/posix.php3.

[33] Mela, M. and Pessi, P. Media Streaming, US Patent Application 10/355234. January 2003.

[34] Mela, M. and Vehmanen, K. and Jalava, T. Method and Apparatus for Utilizing Bluetooth for WLAN Service Authentication and Discovery, US Patent Application 10/880363. June 2004.

[35] Mentor Graphics. Nucleus: Real Time Operating System, 2003. http://www.atinucleus.com/.

[36] Mobilian Corporation. *2.4 GHz and 5 GHz WLAN: Competing or Complementary?*, 2001. http://www.oneunified.net/pdf/Architecture.pdf.

[37] J-C. Monfret, M. Linimon, and L. Uhres. Comp.Realtime News Group Frequently Asked Questions, 2003. http://www.realtime-info.be/encyc/publications/faq/rtfaq.htm.

[38] Motorola, Inc. Motorola Adds Texas Instruments OMAP Processor for Voice Over IP Wireless LAN Capabilities, June 2003. http://www.motorola.com/mediacenter/news/detail/0,1958,2934_2394_23,00.html.

[39] Jussi Mutanen. *A System for Real-time High-quality Audio Streaming*. Helsinki University of Technology, 2002.

[40] Dana Myers. *Developing Speech Applications for Personal Handheld Devices on TI's OMAP Platform*. Texas Instruments, Inc., October 2002. White Paper.

[41] Nokia, Inc. *Mobile Internet Technical Architecture, 3: Visions and Implementations*. IT Press, Ltd., July 2002.

[42] Nokia, Inc. Nokia and Texas Instruments Expand Agreement to Boost Smart Phone Market, February 2003. http://press.nokia.com/PR/200302/892169_5.html.

[43] Nokia Research Center. DSP Gateway for OMAP 1510, 2006. http://dspgateway.sourceforge.net/.

[44] Nokia Research Center. Sofia-SIP, a RFC3261 compliant SIP User-Agent library, 2006. http://sofia-sip.sourceforge.net/.

[45] Parameshwar, N. and Aravamudhan, L. and Reece, C. *Advanced SIP Series: SIP and 3GPP Operations*. Award Solutions, Inc., 2002. http://www.sipcenter.com/files/Award_Advanced_SIP_3GPP2.pdf.

[46] Parameshwar, N. and Reece, C. *Advanced SIP Series: SIP and 3GPP*. Award Solutions, Inc., 2002. http://www.sipcenter.com/news/sipcognition/Award_SIP_3GPP-2.pdf.

[47] David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware/Software Interface*. Morgan Kaufmann Publishers, Inc., 2 edition, 1998.

[48] L. Peeters, B. Rodiers, R. Swennen, and K. Van Caekenberghe. Wireless LAN: Protocols & Vendors, April 2002. http://www.esat.kuleuven.ac.be/ h239/reports/2001/wlan/analysis.php.

[49] Perceptual Evaluation of Speech Quality (PESQ). PESQ home page, 2005. http://www.pesq.org/.

[50] C. Perkins, O. Hodson, and V. Hardman. A Survey of Packet Loss Recovery Techniques for Streaming Audio. *IEEE Network*, 12(5):40–48, September 1998.

[51] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J.C. Bolot, A. Vega-Garcia, and S. Fosse-Parisis. RTP Payload for Redundant Audio Data. RFC 2198, Internet Engineering Task Force, September 1997.

[52] Colin Perkins. *RTP Audio and Video for the Internet*. Addison-Wesley, 2003.

[53] Pumpkin, Inc. *Salvo User Manual*, 3.2.2 edition, August 2003. http://www.microchip.com/download/tools/picmicro/code/third/salvousr220.pdf.

[54] Pumpkin, Inc. Salvo, 2004. http://www.pumpkininc.com/.

[55] Quadros Systems, Inc. RTXC Quadros: Real Time Operating System, 2003. http://www.rtxc.com/.

[56] Real Time Microsystems. SPARK, 2003. http://www.realtimemicrosystems.com/.

[57] J. Rosenberg and H. Schulzrinne. An RTP Payload Format for Generic Forward Error Correction. RFC 2733, Internet Engineering Task Force, December 1999.

[58] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with the Session Description Protocol (SDP). RFC 3264, Internet Engineering Task Force, June 2002.

[59] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 2543, Internet Engineering Task Force, June 2002.

[60] Sanneck, H. and Le, N. and Haardt, M. and Mohr, W. Selective packet prioritization for wireless Voice over IP. In *Fourth International Symposium on Wireless Personal Multimedia Communication*, pages 621–630, September 2001. http://sanneck.net/research/publications/papers/Sann0109_Wireless-VoIP.pdf.

[61] Sanneck, H. and Le, N. T. L. Speech-property-based FEC for Internet Telephony Applications. In *Multimedia Computing and Networking 2000*, volume 3969, pages 38–51. The International Society for Optical Engineering, SPIE-Int. Soc. Opt. Eng, 2000.

[62] H. Schulzrinne. SIP Service Providers, July 2003. http://www1.cs.columbia.edu/sip/service-providers.html.

[63] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 3551, Internet Engineering Task Force, July 2003.

[64] H. Schulzrinne, S. Casner, R. Frederick, and Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Obsoleted by RFC 3550), Internet Engineering Task Force, January 1996.

[65] H. Schulzrinne, S. Casner, R. Frederick, and Jacobson V. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force, July 2003.

[66] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 1889, Internet Engineering Task Force, April 1998.

[67] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and A. Narasimhan. Real Time Streaming Protocol 2.0 (RTSP). Work in Progress (Update to RFC 1889), Internet Engineering Task Force, March 2006. draft-ietf-mmusic-rfc2326bis-12.txt.

[68] Jari Selin. *Media Management in IP Telephony Systems*. Helsinki University of Technology, 2001.

[69] H. Shi. Acoustics, Speech, and Signal Processing. In *ICASSP '00. Proceedings*, volume 6, pages 3211 – 3214. IEEE International Conference on, June 2000.

[70] Skype. Skype, 2006. http://www.skype.com/.

[71] Sonera. Sonera Puhekaista, 2002. http://www.puhekaista.sonera.fi/.

[72] J. Spencer. Converged Platforms Get Real. *Telecommunications*, pages 59–60, October 2001.

[73] Standard Performance Evaluation Corporation, 2003. http://www.spec.org/.

[74] Standard Performance Evaluation Corporation. Spec cpu2000, 2003. http://www.spec.org/cpu2000/.

[75] Stevens, Richard W. *UNIX Network Programming*, volume 1. Prentice Hall, 2 edition, 1998.

[76] Texas Instruments, Inc. *OMAP5910 Dual Core Processor Data Manual*, August 2002. White Paper, Literature Number SPRS197.

[77] Texas Instruments, Inc. Texas Instruments and Ericsson Cooperate on a DSP-based Open Multimedia Applications Platform for Next Generation Wireless Handsets, January 2002. http://www.ti.com/sc/docs/news/2000/00001.htm.

[78] Texas Instruments, Inc., 2003. http://www.ti.com/.

[79] Texas Instruments, Inc. *OMAP5910 Dual-Core Processor*, November 2003. Literature Number: SPRS197C.

[80] Texas Instruments, Inc. Real-Time OS: DSP/BIOS: Other DSPs RTOS, 2003. http://dspvillage.ti.com/.

[81] A. Tsukamoto. RTP QoS Measurement Tool, 2002. http://www1.cs.columbia.edu/ kns10/projects/spring2002/rtpqos/.

[82] Kai Vehmanen. *Design of Low Latency Audio Software for General Purpose Operating Systems*. University of Turku, 2002.

[83] M. Westerlund, A. Lakaniemi, and Q. Xie. Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs. RFC 3267, Internet Engineering Task Force, June 2002.

[84] S. Whitehead, M-J. Montpetit, and X. Marjou. An Evaluation of Session Initiation Protocol (SIP) for use in Streaming Media Applications. Work in progress, Internet Engineering Task Force, February 2006. draft-whitehead-mmusic-sip-for-streaming-media-00.txt.

[85] Wikipedia, The Free Encyclopedia. Millions Instructions Per Second, March 2006. http://www.wikipedia.org/wiki/Million_Instructions_Per_Second.