HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Panu Ranta

Adapting media elements of MMS messages using digital signal processor

Master of Science Thesis

HELSINKI UNIVERSITY OF TECHNOLOGY | **ABSTRACT OF THE MASTER'S THESIS**

Author: Panu Ranta

Name of the thesis: Adapting media elements of MMS messages using digital signal processor

Date: 6 November 2003 | **Number of pages:** 53

Department: Department of Computer Science and Engineering
Professorship: T-106

Supervisor: Professor Eljas Soisalon-Soininen
Instructor: Teemu Himanen, M.Sc. (Tech.)

The last decade of the previous century saw the explosive growth in the usage of mobile telephones and the Internet. During the decade mobile telephones evolved in the industrialised nations from exclusive items to almost everyday commodities. Mobile telephones are not used only for speech services anymore, as text messaging has become a popular way to communicate, especially among teenagers. Companies have introduced a wide variety of devices for mobile communication designed to fulfil the diverse needs of consumers.

Mobile networks and devices will continue to evolve. Increasing transmissions speeds and more powerful mobile devices will enable the introduction of new services. One example of these new services is the multimedia messaging service that allows users to include media elements like images and video to text messages. To guarantee that people using different mobile devices will be able to communicate with each other, these media elements need to be adapted for each device. Digital signal processors are designed to efficiently process media elements.

This thesis introduces different mobile networks and describes how they have evolved. Multimedia messaging service is described in detail as an example of the new services that introduce the need for media adaptation. Different adaptation cases and the need for adaptation are discussed. Widely used media formats are also briefly introduced. Basics of image coding are given and the JPEG image format is thoroughly investigated. Digital signal processors are described as well as the simple application that was implemented to study closer the image processing algorithms and the processors.

Keywords: digital signal processor, JPEG, multimedia messaging service

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä: Panu Ranta

Työn nimi: MMS-viestien mediaelementtien mukauttaminen digitaalisella signaaliprosessorilla

Päivämäärä: 6.11.2003 **Sivumäärä:** 53

Osasto: Tietotekniikan osasto
Professuuri: T-106

Työn valvoja: Professori Eljas Soisalon-Soininen
Työn ohjaaja: DI Teemu Himanen

Matkapuhelinten ja Internetin käyttö yleistyi räjähdysmäisesti edellisen vuosisadan viimeisellä vuosikymmenellä. Matkapuhelimista tuli teollisuusmaissa lähes arkisia kulutustavaroita. Nykyään matkapuhelimia ei käytetä ainoastaan puhumiseen, vaan tekstiviesteistä on tullut suosittu viestintämuoto erityisesti nuorten keskuudessa. Yritykset ovat esitelleet laajan valikoiman erilaisia laitteita langattomaan viestintään tarkoituksenaan tyydyttää kuluttajien moninaisia tarpeita.

Matkapuhelinverkot ja -puhelimet tulevat kehittymään myös jatkossa. Entistä tehokkaammat kannettavat viestintälaitteet sekä niiden tiedonsiirtonopeuksien kasvu mahdollistavat uusien palveluiden synnyn. Esimerkki tällaisista uusista palveluista on multimediaviestipalvelu, joka mahdollistaa mediaelementtien, kuten kuvien ja äänien, liittämisen tekstiviesteihin. Jotta ihmiset voisivat viestiä keskenään käyttäen erilaisia viestintälaitteita, niin mediaelementtejä tulee mukauttaa laitteisiin sopiviksi. Digitaaliset signaaliprosessorit on suunniteltu käsittelemään mediaelementtejä tehokkaasti.

Tämä diplomityö esittelee erilaisia matkapuhelinverkkoja sekä niiden kehittymistä. Esimerkkinä mediaelementtien mukauttamista vaativista palveluista tarkastellaan multimediaviestipalvelua. Erilaisia mediaelementtien mukautuksia sekä tarvetta mukauttamiselle pohditaan. Yleisesti käytettyjä mediaformaatteja esitellään lyhyesti. Kuvakoodauksen perusteet käsitellään ja JPEG-kuvaformaattiin tutustutaan perusteellisesti. Digitaalisten signaaliprosessoreiden tyypillisiä ominaisuuksia perustellaan. Kuvien mukauttamiseen käytettävien algoritmien lähempää tutkimista varten toteutettu ohjelma testausympäristöineen kuvaillaan.

Avainsanat: digitaalinen signaaliprosessori, JPEG, multimediaviestipalvelu

Preface

This thesis has been written in the Signal Processing Systems competence area of Nokia Networks. I would like to thank my instructor Teemu Himanen for his continuous support and encouragement. I am also grateful to all the other people who have helped me. Finally, I am going to congratulate myself, I have proven the impossible really exists.

Helsinki, 6 November 2003

Panu Ranta

Table of contents

ABSTRACT OF THE MASTER'S THESIS II
DIPLOMITYÖN TIIVISTELMÄ III
PREFACE IV
TABLE OF CONTENTS V
ABBREVIATIONS VII
1 INTRODUCTION 1
1.1 Background 1
1.2 Scope of the thesis 2
1.3 Structure of the thesis 2
2 3GPP NETWORK 3
2.1 Mobile networks and standardisation 3
2.2 Global System for Mobile Communications 4
2.2.1 High Speed Circuit Switched Data 5
2.2.2 General Packet Radio Service 5
2.2.3 Enhanced Data rates for GSM Evolution 6
2.3 Universal Mobile Telecommunications System 7
2.4 Multimedia messaging service 8
2.4.1 Overview 8
2.4.2 Architecture 9
2.5 Media adaptation needs 12
3 MMS MESSAGES 13
3.1 Message structure 13
3.1.1 Multi-purpose Internet Mail Extensions 13
3.1.2 Synchronised Multimedia Integration Language 15
3.2 Media formats 16
3.2.1 Image formats 17
3.2.2 Audio formats 18
3.2.3 Video formats 20
3.3 Media adaptation 21
4 IMAGE CODING AND JPEG 23
4.1 Basics of image coding 23
4.1.1 Colour spaces 24
4.1.2 Compression 25
4.2 Discrete cosine transform 27
4.3 Huffman coding 29
4.4 JPEG image format 30
4.4.1 Overview 30
4.4.2 Encoding and decoding example 31
5 JPEG ADAPTER 33
5.1 Features and reasoning 33
5.2 Digital signal processors 35
5.2.1 Overview 35
5.2.2 Used digital signal processor 36
5.3 Computing discrete cosine transform 38
5.4 Implementation 39
5.4.1 Overview 39
5.4.2 Operating system 40

5.4.3 IP stack 40
5.4.4 JPEG library 41
6 TESTING AND MEASUREMENTS 43
6.1 JPEG adapter tester 43
6.2 Media Gateway 44
6.3 Test environment 46
6.3.1 Overview 46
6.3.2 Transcoding plug-in unit 47
6.4 Measurements 48
6.4.1 Overview 48
6.4.2 Results and analysis 49
6.4.3 Future measurements 52
7 CONCLUSIONS 53
REFERENCES 54

Abbreviations

3GPP	Third Generation Partnership Project
AAC	Advanced Audio Coding
AMPS	Advanced Mobile Phone Service
AMR	Adaptive Multi-Rate
ATM	Asynchronous Transfer Mode
BMP	Windows Bitmap
BPP	Bits Per Pixel
CDMA	Code Division Multiple Access
CSD	Circuit Switched Data
DCT	Discrete Cosine Transform
DSP	Digital Signal Processor
EDGE	Enhanced Data rates for GSM Evolution
EGPRS	Enhanced GPRS
EMS	Enhanced Messaging Service
ETSI	European Telecommunications Standards Institute
FFT	Fast Fourier Transform
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GPP	General-Purpose Processor
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HSCSD	High Speed Circuit Switched Data
HSDPA	High Speed Downlink Packet Access
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IJG	Independent JPEG Group
IMT-2000	International Mobile Telecommunications 2000
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group
MAC	Multiply and Accumulate
MIDI	Musical Instrument Digital Interface
MIME	Multi-purpose Internet Mail Extensions
MIPS	Million Instructions Per Second
MJPEG	Motion JPEG
MMS	Multimedia Messaging Service
MMSC	MMS Centre
MPEG	Moving Picture Experts Group
NMT	Nordic Mobile Telephone
OMA	Open Mobile Alliance
PCM	Pulse Code Modulation
PDC	Personal Digital Cellular
PNG	Portable Network Graphics

PSTN	Public Switched Telephone Network
QOS	Quality Of Service
RFC	Request For Comments
SMIL	Synchronised Multimedia Integration Language
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SP-MIDI	Scalable Polyphony MIDI
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
TDM	Time-Division Multiplexing
TDMA	Time Division Multiple Access
TIA	Telecommunications Industry Association
TPU	Transcoding Plug-in Unit
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WCDMA	Wideband Code Division Multiple Access
WSP	Wireless Session Protocol
WWW	World Wide Web
XML	Extensible Markup Language

1 Introduction

1.1 Background

Mobile telephones and the Internet have changed the way people communicate more than anything since the invention of the telephone over a century ago. The success of these new communication methods is evident. During the last decade the number of Internet users increased from a couple of million to 500 million and the number of mobile telephone subscribers increased from ten million to almost one billion [ITU].

Despite the success of mobile telephones and the Internet, the evolution of communication networks is not over but rather has just begun. Many people would like to see a future where people use different communication services whenever and wherever they want regardless of the network that provides the services. The biggest challenge of the telecommunications industry in this decade is to bring the different networks together and build one truly interworking communication network.

Advances in electronics will provide the basis for new communication devices. Ever increasing computing power, larger memories, and faster data transfer capabilities allow manufactures to design more diverse devices. Mobile telephones are used today mainly for speech and text services but in the future mobile devices will be used to access much more versatile services. One of the new services offered in mobile networks is the multimedia messaging service (MMS) that allows users to send and receive messages, which are composed from different media elements such as text, image, audio, and video.

New devices and services will also introduce new problems and challenges. Devices will differ greatly in capabilities and services will impose strict requirements for them. Interoperability will be the key to success. Service providers must ensure that people will be able to utilise the services using different devices. Media content must be adapted to match the capabilities of the devices. Media processing requires a lot of computing resources, which are scarce on battery-operated mobile devices and in many cases the adaptation must be done in the network connecting the devices.

Digital signal processors (DSP) are used to adapt speech from one format to another in current mobile networks. DSPs are designed to perform certain signal processing

algorithms much faster than general-purpose processors, which are used, for example, in PCs. Same or similar algorithms that are needed in speech processing are also needed when processing images, audio in general, or video. Therefore, it seems natural to use DSPs to adapt media content also in future communication networks.

1.2 Scope of the thesis

The aim of this thesis is to study the evolving mobile networks and services that need media adaptation. The focus is on the multimedia messaging service and JPEG image format. Reasons are given why media adaptation should be done using a digital signal processor. Doing the adaptation in a network is emphasised but much of the discussion apply also to adaptations done in mobile devices. Adaptation of the entire MMS messages is out of the scope of this thesis and the discussion is limited to adaptation of the media elements inside MMS messages. Streaming services and adaptation of media streams are also out of the scope of this thesis.

1.3 Structure of the thesis

Chapter 2 lists different mobile networks and describes the architectures and services of GSM and UMTS networks. The multimedia messaging service is introduced in detail. Chapter 3 describes the structure and contents of MMS messages. Different media formats that are commonly used in MMS messages are briefly presented and media adaptation is defined and discussed. Chapter 4 introduces general image coding principles using the JPEG image format as an example. Compression algorithms used in JPEG are studied closely. Chapter 5 contains the implementation part of this thesis. Used hardware and software are introduced and the implemented program is motivated and described. Chapter 6 describes the test environment and the measurements that were conducted. Measurement results are presented and analysed. Finally, conclusions are presented in Chapter 7.

2 3GPP network

This chapter describes the architecture and services of the 3GPP network, which is the most widely used mobile communication network in the world. First, different mobile networks and standardisation organisations are listed. The second and third sections describe GSM and UMTS networks. The most important network elements are presented and different services are introduced. The fourth section describes multimedia messaging service (MMS) in more detail. Related network elements and signalling flows are examined. The fifth section discusses the needs for media adaptation in mobile networks.

2.1 Mobile networks and standardisation

The evolution of mobile networks is often divided into three generations. First generation means analogue networks like Advanced Mobile Phone Service (AMPS) that was developed in the USA and Nordic Mobile Telephone (NMT) that was developed in Europe. Second generation means digital networks like the Global System for Mobile Communications (GSM) that is used in most parts of the world, Personal Digital Cellular (PDC) that is used in Japan, IS-95 that is used mainly in the USA and South Korea, and IS-136 that is used in the Americas. IS-95 is also known as Code Division Multiple Access (CDMA) after its radio interface and IS-136 is known as Time Division Multiple Access (TDMA) for the same reason. They are both specified by the Telecommunications Industry Association (TIA).

GSM was originally specified by the European Telecommunications Standards Institute (ETSI). In 1998 ETSI and four other standardisation organisations founded the Third Generation Partnership Project (3GPP) to specify a third generation mobile network based on evolved GSM core network and Wideband Code Division Multiple Access (WCDMA) radio interface. This network is called Universal Mobile Telecommunications System (UMTS). One of the reasons why a third generation mobile network was designed was an attempt to create a global standard. GSM standardisation was moved from ETSI to 3GPP in the year 2000.

ETSI is a member of the International Telecommunication Union (ITU), which is a telecommunications agency of the United Nations. One of the purposes of ITU is to

regulate globally the use of radio frequencies. ITU calls third generation mobile networks as International Mobile Telecommunications 2000 (IMT-2000). It has approved five IMT-2000 radio interfaces including CDMA2000 and WCDMA. CDMA2000 network is based on IS-95 and it is standardised by 3GPP2. Despite similar names WCDMA and CDMA2000 are not compatible.

2.2 Global System for Mobile Communications

In the early 1980s a need for a single European mobile communication system arose. Many incompatible analogue first generation networks existed but it was seen that a single digital mobile network would better serve the needs of people and business. Global System for Mobile Communications (GSM) was standardised during the 1980s and GSM Phase 1 was finalised in 1990 [Hil01]. The first network was launched commercially two years later. The second major milestone in GSM standardisation was GSM Phase 2 in 1995. Yearly GSM Phase 2+ releases followed after that.

A simplified overview of GSM architecture is presented in Figure 2.1. A GSM system is divided into subsystems that are interconnected using well-specified interfaces. Subsystems are further divided into network elements. Standards define logical network elements while real network elements are often combinations of one or more logical network elements. Radio interface between Mobile Station and Base Transceiver Station (BTS) is the distinctive interface of mobile networks.

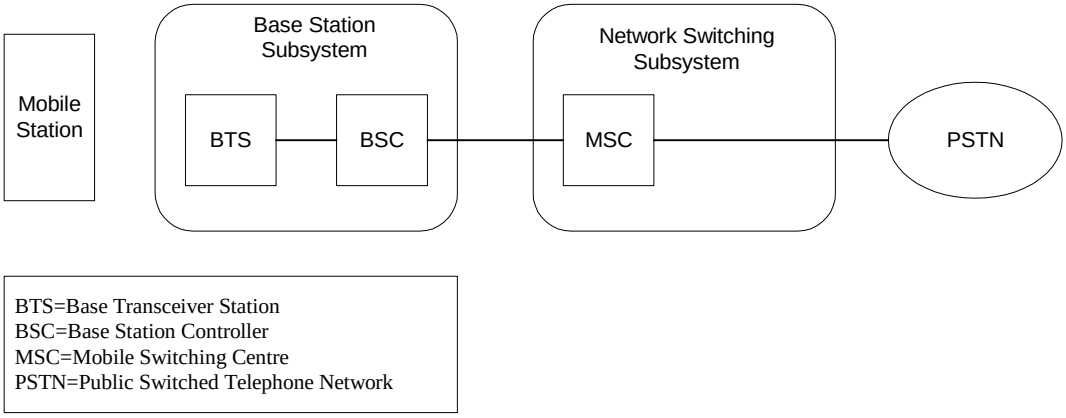


Figure 2.1 Simplified GSM architecture.

The three most important services that GSM provides are speech, Circuit Switched Data (CSD), and Short Message Service (SMS). In traditional wireline telephone network, Public Switched Telephone Network (PSTN), Pulse Code Modulated (PCM) speech is

transported using 64 kbit/s. GSM uses Full Rate speech codec to compress speech to 13 kbit/s, Enhanced Full Rate to 12.2 kbit/s, or Half Rate to 5.6 kbit/s. CSD is the oldest data service GSM offers. It was introduced in the mid-1990s with the speed of 9.6 kbit/s.

SMS was introduced soon after the first GSM networks were launched [3GPP 23.040], [Pee00]. It provides a way to send short text messages between mobile stations. Message length is 140 octets, usually 160 characters. SMS follows a store-and-forward service model where a sent SMS message is stored to an SMS centre until it can be pushed to the receiving mobile station. This differs from the traditional e-mail service where a client fetches messages from a server.

2.2.1 High Speed Circuit Switched Data

GSM uses Time Division Multiple Access (TDMA) to share the resources in the radio interface. Time is divided into small slots and each mobile station is allowed to send or receive data during its timeslot. Nowadays one timeslot can be used to transmit data at the maximum speed of 14.4 kbit/s. High Speed Circuit Switched Data (HSCSD) is a method to use more than one timeslot to achieve greater transmission speeds [3GPP 22.034], [3GPP 23.034]. It was introduced in the late 1990s and in 2002 HSCSD was offered in about 40 networks. Up to four timeslots can be used, which means the maximum total data rate is 57.6 kbit/s. Usually different number of timeslots are used for sending and receiving. The data rate and delay vary depending on the number of transmission errors on the radio interface. Constant data rate and delay can be achieved if no link layer protocol is used. This is called transparent data service and it can be used for services that tolerate errors better than delay. Video call is one example.

2.2.2 General Packet Radio Service

The success of the Internet and the need for higher data transmission speeds led to the introduction of General Packet Radio Service (GPRS) in the year 2000 [3GPP 22.060], [3GPP 23.060]. Three years later GPRS was available in almost 200 out of the over 500 GSM networks in the world. This extension to the GSM system provides a packet switched data service to mobile stations. Circuit switched service allocates a circuit and other resources for the duration of the call. Packet switched service uses small packets to transfer data. No circuit is reserved and resources are only used when data is actually being transmitted.

Changes to the original GSM architecture are shown in Figure 2.2. Due to the changes in the radio interface, GPRS requires new mobile and base stations. The overall architecture remains the same but a new core network, GPRS Packet Core, with two new network elements is introduced. It can provide a connection to the Internet or to some other packet data network.

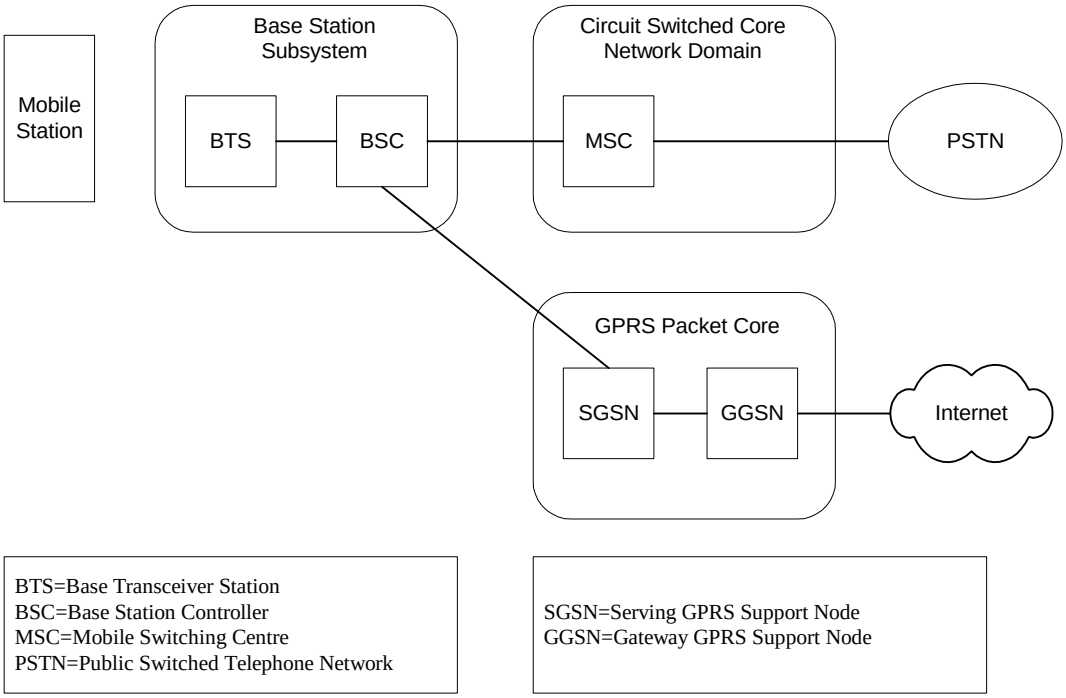


Figure 2.2 GPRS architecture.

Used channel coding scheme (CS) and the number of timeslots determine the speed of the GPRS service [3GPP 43.064]. Channel coding means adding error detection and correction information to the data to being transmitted over the radio interface. The maximum speed of one timeslot is 21.4 kbit/s when no error correction is used (CS-4). One mobile station can use up to eight timeslots and the maximum total data rate is 171.2 kbit/s. In 2003 a common speed of one timeslot in real networks is 13.4 kbit/s (CS-2) and most mobile stations can use three or four timeslots. Delays in GPRS networks are considerably higher than in fixed data networks.

2.2.3 Enhanced Data rates for GSM Evolution

Changing the channel coding scheme is one way to increase the transmission speed as was explained in the previous section. Another way is to change the modulation technique. Modulation means altering a property of an analogue signal, for example, to represent changes in a digital signal. The basic properties of an analogue signal are

amplitude, frequency and phase. GSM uses a phase modulation technique. Enhanced Data rates for GSM Evolution (EDGE) is an evolutionary improvement of the GSM system that introduces nine new channel coding schemes and a new phase modulation technique, which triples the speed of one timeslot [3GPP 43.064]. Advances in electronics and increased processing power in mobile devices have enabled the use of the new channel coding and modulation techniques. EDGE can be applied to both circuit switched data, Enhanced CSD (ECSD), and to packet switched data, Enhanced GPRS (EGPRS). EDGE is also one of the five radio interfaces that ITU has approved for IMT-2000. The first EDGE networks were launched commercially in 2003.

ECSD does not aim to increase the maximum transmission speed but it can use radio resources more efficiently than CSD. The maximum speed of one timeslot is 43.2 kbit/s and the maximum total data rate of 57.6 kbit/s can be achieved using only two timeslots instead of four. The maximum speed of one EGPRS timeslot is 59.2 kbit/s and the maximum total data rate of eight timeslots is 473.6 kbit/s. The first EGPRS networks provide speeds of about 100-150 kbit/s, which is three times faster than GPRS.

2.3 Universal Mobile Telecommunications System

Standardisation work for the successor of GSM began even before the first GSM network was launched but most of the work was done in the latter part of the 1990s. The first version of the Universal Mobile Telecommunications System (UMTS) specification is called Release 1999 and it was published in the year 2000 [Kaa01]. The first commercial UMTS network was launched two years later. The biggest difference between UMTS and GSM systems is the new radio interface based on Wideband Code Division Multiple Access (WCDMA). The core network of UMTS is based on the evolved GSM core network [3GPP 23.002]. The architecture of the UMTS network is shown in Figure 2.3. 3GPP calls UMTS base station as Node B.

The second release of the UMTS specification is called Release 4 and it was ready in 2001. It introduces more drastic changes to the core network. In Release 4 user plane (speech, data) and control plane (signalling) are separated, and they are routed independently. This enables more flexible capacity increases and it encourages competition between network element manufacturers. Release 5 was ready in 2002 and it introduces again major architectural changes. It is based on the popular vision that all traffic will be transported over IP in the future. Release 5 introduces also High Speed

Downlink Packet Access (HSDPA), which further increases data rates at the radio interface.

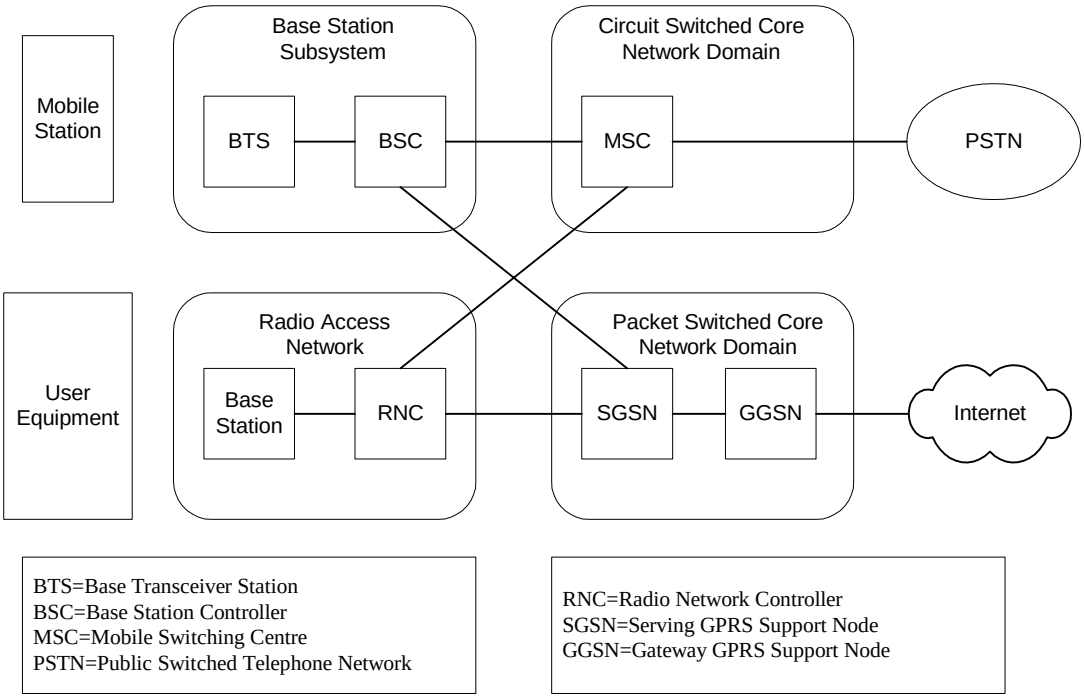


Figure 2.3 UMTS architecture (3GPP Release 1999).

UMTS services and service capabilities are described in [3GPP 22.105]. Quality Of Service (QOS) classes are defined for different services. They set limits for different service parameters like delay and data rate. Supported data rates vary from 144 kbit/s in rural outdoor radio environment to 2048 kbit/s in indoor radio environment. The first UMTS networks provide speeds close to the lower end of the range.

2.4 Multimedia messaging service

2.4.1 Overview

SMS is a hugely successful service and it has led to the development of more advanced messaging services [Teo03], [Rod03]. Some companies have developed proprietary extensions to SMS but the most widely supported extension is the 3GPP's Enhanced Messaging Service (EMS). It supports sending of objects like formatted text, small pictures, and sounds over SMS. EMS is a temporary messaging solution because SMS messages are still being transmitted over signalling links, which are not designed to handle large amounts of traffic efficiently. EMS supports only a limited set of media formats and there is a clear need for a more flexible and expandable messaging service.

Multimedia messaging service (MMS) has been developed by 3GPP to meet the needs of a new messaging service. MMS follows the same store-and-forward service model as SMS into which it tries to incorporate the best features of the Internet e-mail. This makes MMS more attractive to users than SMS but also new problems will arise. MMS allows users to use different media elements to compose multimedia messages, which can be then sent to other users as easily as SMS messages. MMS messages are described in detail in Chapter 3. Use of different media elements and formats introduces interoperability problems that are not present in SMS. Ease of use and interoperability are the key requirements for the success of MMS.

The first version of the MMS specification was a part of the 3GPP Release 1999 but MMS is not designed to be specifically a UMTS service. It can also be used in, for example, GSM networks using different data bearers like GPRS, CSD or SMS. Requirements for MMS are described in [3GPP 22.140] and functional description is in [3GPP 23.140]. 3GPP2 has its own set of specifications for MMS, but they are mostly copied from 3GPP. In 2002 nearly 200 companies formed the Open Mobile Alliance (OMA) whose mission includes ensuring seamless application interoperability. One of the principles of OMA is that the application layer is bearer independent, which means that the same services can be used in UMTS and CDMA2000 networks, for example. OMA has produced an MMS specification suite that complies with the 3GPP specifications.

2.4.2 Architecture

MMS introduces two new logical network elements: MMS Relay and MMS Server. The interface between these two elements is not specified and they are practically always combined into a single network element called the MMS Centre (MMSC). The main tasks for MMSC are storage and handling of MMS messages, and generation of charging data. MMS can be used in many different networks but a common way to connect an MMSC to a GSM network is to use the Wireless Application Protocol (WAP) and GPRS as a bearer for WAP [OMA arch]. Changes to the basic GPRS architecture are shown in Figure 2.4.

One of the design principles of MMS has been the reuse of existing protocols. Especially IP based protocols, specified by the Internet Engineering Task Force (IETF), are used extensively. Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer

Protocol (SMTP) are used for message transport. File Transfer Protocol (FTP) can be used for transport of billing and statistical information. These protocols run on top of Transmission Control Protocol (TCP), which offers a reliable connection-oriented service on IP networks. Second generation mobile networks are not based on IP and a gateway is needed when an MMSC is connected to such network. An existing network element called WAP Gateway can be used if WAP is used in the mobile network and in mobile stations. Connections from mobile stations to WAP Gateway are based on the Wireless Session Protocol (WSP) and connections from WAP Gateway to MMSC are based on HTTP [RFC 2616]. WSP [WAP WSP] is basically a binary form of HTTP that is used in wireless networks. WAP Gateway handles the translation between these two protocols.

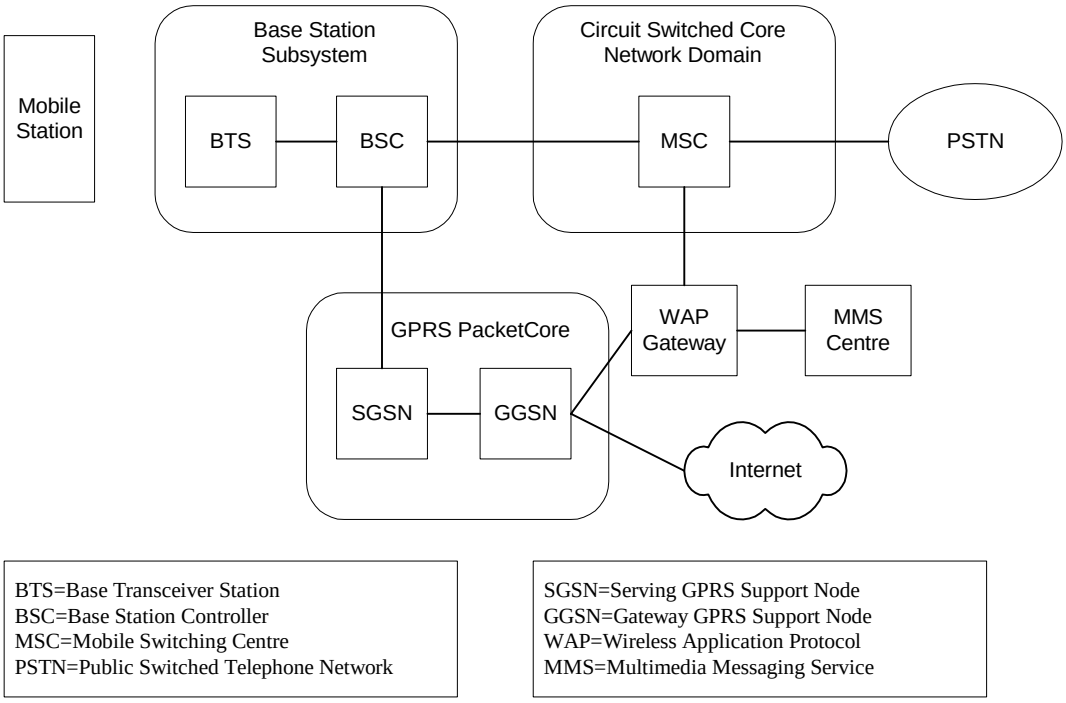


Figure 2.4 Architecture of WAP implementation of MMS.

Signalling related to the sending of an MMS message is shown in Figure 2.5. The MMS Client (mobile station) initiates the sending by sending a send request to the MMSC [OMA ctr]. The actual MMS message is transported along the request. When the MMSC receives the request it sends a confirmation back to the MMS Client and stores the message for retrieval. The MMSC sends also a notification of an incoming MMS message to the receiving MMS Client. The notification is sent using WAP Push, which means that the Push Access Protocol is used to send the notification from the MMSC to

the WAP Gateway, which acts as a Push Proxy Gateway and forwards the notification using SMS. The notification contains the address of the MMS message and the receiving MMS Client uses WSP GET to fetch the actual MMS message after receiving the notification. The SMS message, which contains the notification, is not shown to the user of the mobile station but rather the MMS message is fetched automatically. When the MMSC receives the fetching request it sends the actual MMS message to the receiving MMS Client in a retrieve confirmation message.

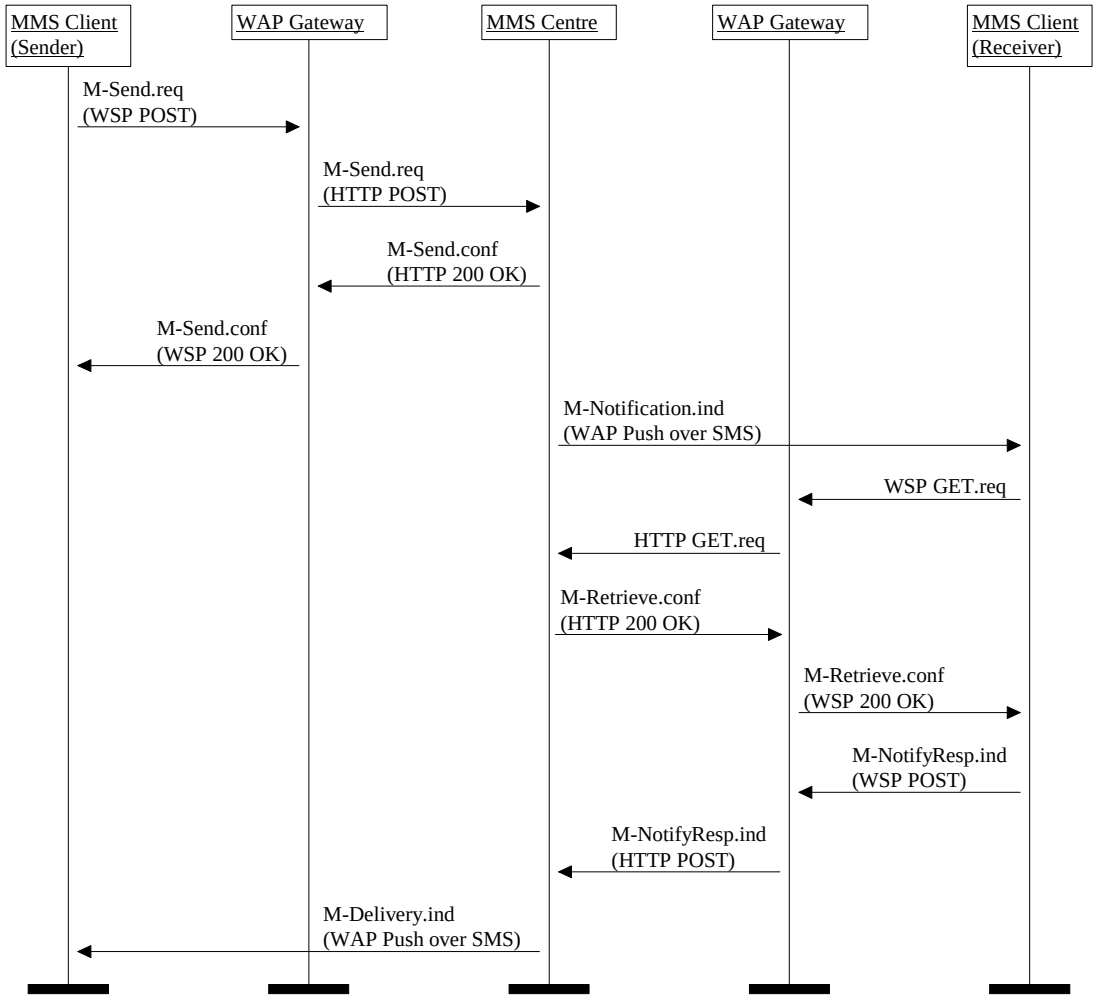


Figure 2.5 Signalling of successful sending of an MMS message.

After receiving the actual MMS message the receiving MMS Client can send an optional acknowledgement to the MMSC using a notify response message. The MMSC can also send an optional delivery report to the sending MMS Client.

2.5 Media adaptation needs

Media adaptation is needed when two parties communicate with each other using devices capable of presenting an incompatible set of media formats. Different media formats are described in Section 3.2. Incompatibility may be caused by many different reasons. The devices may be of different type like a mobile telephone and a PC. If the devices are of the same type they may be of different product category like a high-end mobile telephone and a low-end mobile telephone. If the devices are of the same type and same product category they may be using different standards that require the use of different media formats like a GSM phone and an IS-95 phone. Also, copyright laws and other legislation differ in many countries and may cause a need for the adaptation of media elements. When introducing new media formats or services, it is important that new devices using the new media formats will able to communicate with the old devices.

When two parties communicate using mobile devices and a network of some kind, the adaptation can be done in a mobile device, in the network, or in both. The decision on where to adapt the media elements depends on many things. The adaptation of the media elements requires a lot of processing power but battery-operated mobile devices can only have limited computing capabilities compared to network elements. A support for a large set of media formats requires large software and, again, mobile devices have limited memory compared to network elements. Also, upgrading the software of customer owned mobile devices is much harder than of network operator owned network elements. Upgrades may be needed because of bug fixes or adding support for new media formats, for example.

Media adaptation is not a new function introduced by UMTS. Second generation mobile networks use media adaptation already to enable phone calls between different networks. For example, in GSM system speech is adapted from codecs like Full Rate and Enhanced Full Rate to PCM when calling to PSTN and vice versa. This real-time adaptation is done in a network element called transcoder. Ever increasing data rates and the introduction of new services like MMS indicate that the need for the adaptation of media elements may soar in coming years.

3 MMS messages

This chapter describes the structure and contents of MMS messages. The first section shows how different media elements are used to compose a static multimedia message or an interactive presentation embedded in an MMS message. The second section divides media formats into three categories and introduces the media formats that MMS clients shall support in order to guarantee interoperability. The third section defines what media adaptation means and introduces a classification of different kind of media adaptations. Adaptations specific to certain media formats are also given as examples.

3.1 Message structure

Protocol reuse is one of the design principles of MMS as was seen in Section 2.4.2. Existing message formats are also reused in MMS. The Multi-purpose Internet Mail Extensions (MIME) are used to describe the structure of an MMS message and the Synchronised Multimedia Integration Language (SMIL) is usually used to describe how an MMS message is presented. Presentation format describes in what order the media elements are shown and how they are positioned on the screen, for example.

An MMS message can be sent either to a normal mobile telephone number or to an Internet e-mail address. Telephone numbers are standardised by ITU in E.164 and e-mail addresses by IETF in [RFC 2822]. An MMSC is addressed by a Uniform Resource Identifier (URI) specified in IETF RFC 2396.

3.1.1 Multi-purpose Internet Mail Extensions

The format of the Internet e-mail messages was originally specified by IETF in RFC 822, which was released in 1982. An updated version of the specification was released in 2001 [RFC 2822]. It specifies that a message is a series of 7-bit US-ASCII characters and that a message consists of header fields and an optional body. This is highly restrictive as the specification does not define how images or other media elements can be included in the messages, or how more advanced character sets can be used, for example. These issues are left out on purpose and they are handled in a series of RFCs that specify MIME.

[RFC 2045] specifies the header fields used to describe the structure of MIME messages and [RFC 2046] specifies the general structure of the MIME media typing system and

defines an initial set of media types. Other RFCs specify how character sets other than US-ASCII can be used in the message headers, how MIME media types are registered, and so on. One of the new header fields is called Content-Type. The purpose of this field is to describe the data contained in the body part of the message. The value of the field is called media type. The MIME media typing system is hierarchical and seven top-level media types have been defined in [RFC 2046] including five discrete media types (text, image, audio, video, and application) and two composite media types (multipart and message).

The discrete media types are discussed in Section 3.2. The multipart composite media type defines how multiple media types can be included in a single message. Subtypes define the relationships between different media types in a message. Four subtypes have been defined in [RFC 2046]: mixed, alternative, parallel, and digest. Subtype related is defined in [RFC 2387].

When an MMS messages is transported in a WSP or HTTP message, the value of the Content-Type field of the encapsulating protocol is application/vnd.wap.mms-message. Subtype vnd stands for vendor. An MMS message consists of two parts: header fields and a body. The header fields follow the format defined in [RFC 2822]. Additional fields are defined in [OMA enca]. The Content-Type field of an MMS message is usually application/vnd.wap.multipart.related. This media type is a compact binary form of the corresponding MIME media type. It and other media types designed for wireless networks are specified by the WAP Forum in [WAP WSP]. One of the media types in a multipart message may be application/smil that defines how the media elements are presented to the user. If the presentation part does not exist, it is up to the MMS client to decide how the MMS message is presented.

An example MMS message is shown in Figure 3.1. The message type is send request and it is encapsulated in a HTTP POST request. This message is transported from a WAP Gateway to an MMSC as can be seen from Figure 2.5. The send request contains the actual MMS message, which in this case contains the SMIL presentation from Figure 3.2 and six media elements that are omitted from the example for simplicity. The first four lines of the example contains the HTTP headers and after an empty lines comes 11 lines of MMS headers followed by an empty line and finally the body of the MMS message.

```
POST / HTTP/1.1
Host: mmsc.operator.com.example:80
Content-Type: application/vnd.wap.mms-message
Content-Length: 49688

X-Mms-Message-Type: m-send-req
X-Mms-Transaction-ID: 001
X-Mms-Version: 1.0
Date: Wed Aug 11 16:55:52 2003
From: +358501234567/TYPE=PLMN
To: +358507654321/TYPE=PLMN
Subject: Hello from Wolrd!
Content-Type: application/vnd.wap.multipart.related;
              type=application/smil;
              start=<presentation-part>;
              boundary="gc0p4Jq0M2Yt08j34c0p"

--gc0p4Jq0M2Yt08j34c0p
Content-ID: <presentation-part>
Content-Type: application/smil

[Example SMIL presentation from Figure 3.2 comes here.]

--gc0p4Jq0M2Yt08j34c0p
Content-ID: <CID1>
Content-Location: FirstText.txt
Content-Type: text/plain

The text of the first slide says Hello World!

--gc0p4Jq0M2Yt08j34c0p
Content-ID: <CID2>
Content-Location: FirstImage.jpg
Content-Type: image/jpeg

[The rest of the example is removed to save some space.]
```

Figure 3.1 MMS message encapsulated in a HTTP POST request.

3.1.2 Synchronised Multimedia Integration Language

SMIL is a language that can be used to construct interactive audiovisual presentations [W3C SMIL]. It is specified by the World Wide Web Consortium (W3C), which was created in 1994 to lead the World Wide Web (WWW) to its full potential. W3C is best known by the Hypertext Markup Language (HTML) that is used to write hypertext documents published in WWW. SMIL tries to capture the simplicity and popularity of HTML. They are both text-based languages that are easy to learn and use. The first version of the SMIL specification was ready in 1998 and the current version, SMIL 2.0, was published in 2001.

Example SMIL presentation is shown in Figure 3.2. In this example, two layout regions and also two slides are defined. Both slides contain an image, a text and an audio clip. All these three elements within a slide are presented in parallel. The media elements are part of the same multipart message as the presentation, which refers to them. Duration of the first slide is eight seconds and the second slide is one second shorter.

```
<smil>
    <head>
        <layout>
            <root-layout width="352" height="144"/>
            <region id="Image" width="176" height="144" left="0" top="0"/>
            <region id="Text" width="176" height="144" left="176" top="0"/>
        </layout>
    </head>
    <body>
        <par dur="8000ms">
            <img src="FirstImage.jpg" region="Image"/>
            <text src="FirstText.txt" region="Text"/>
            <audio src="FirstSound.amr"/>
        </par>
        <par dur="7000ms">
            <img src="SecondImage.jpg" region="Image"/>
            <text src="SecondText.txt" region="Text"/>
            <audio src="SecondSound.amr"/>
        </par>
    </body>
</smil>
```

Figure 3.2 Example SMIL presentation from [OMA conf].

3.2 Media formats

In this thesis a media element means an entity that is specified by a media format and is interpreted by a communicating party. Basically a media element is a file that is not executed. Some media formats are specified by standardisation organisations, some by companies, and some media formats are not specified at all. A media format specification describes usually the algorithms and the file format that are used.

Media formats can be categorised in many ways. The MIME media typing system was introduced in Section 3.1.1. A top-level MIME media type describes the general type of the media format and subtypes describe the specific media formats. The top-level type text describes plain text and different kinds of formatted text formats. The adaptation of text is usually not done using a digital signal processor and it is out of the scope of this thesis. The top-level types image, audio, and video are described in detail in the next sections. The fifth and last discrete top-level media type is application, which is used to describe all other media formats.

The categorisation of the media formats is not strict. Many times a video format specification includes also a specification for an audio track in addition to the video track. A video file without a video track is often very similar to an audio file. Also, from the user's point of view the media format is not as essential as the presentation format. SMIL can be used to construct a slide show, a sequence of discrete images, with an audio track that can look exactly like a video file, for example.

MMS does not restrict in any way the media formats included in MMS messages. Hundreds of different MIME media subtypes have been defined and the subtype application/octet-stream is used to describe arbitrary binary data. This leads easily into compatibility problems and therefore 3GPP has specified in [3GPP 26.140] a minimum set of media formats that all MMS clients supporting a specific media type shall support. Full support of a media format is usually not required but a certain baseline or a subset of features is enough.

The next three sections give an overview on image, audio, and video media types. Media formats from [3GPP 26.140] are briefly described and also other widely used or otherwise important formats are mentioned. References to the specifications of these media formats can be found from [3GPP 26.140].

3.2.1 Image formats

Dozens of different image formats have been developed for different purposes. No single image format is optimal for all kind of images and usually one format is optimised for certain image types. Image coding is described in detail in Chapter 4.

3.2.1.1 JPEG

Joint Photographic Experts Group (JPEG) is the most widely used image format in the Internet. It is designed to compress continuous-tone still images, photographs, and a typical compression ratio of 20:1 can be achieved without noticeable changes in quality. Baseline JPEG does not require any licensing fees, which is one of the reasons for its success. JPEG image format is described in detail in Section 4.4.

3.2.1.2 GIF

Graphics Interchange Format (GIF) was specified by a company called CompuServe in the late 1980s. GIF uses lossless dictionary-based LZW compression algorithm. Typical compression ratio is about 2:1. The amount of colours of a GIF image is limited to 256 and GIF is best suited for line drawings, logos and other non-realistic images. GIF is widely used in WWW but patent disputes and more advanced image formats are making GIF obsolete.

3.2.1.3 PNG

Portable Network Graphics (PNG) is an advanced patent-free replacement of GIF. PNG is lossless like GIF but it compresses slightly better. PNG was designed in the latter part

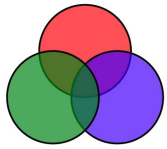
of the 1990s and the specification has been published as IETF RFC 2083, later as a W3C Recommendation, and it is also being standardised by International Organization for Standardization (ISO).

3.2.1.4 SVG

Scalable Vector Graphics (SVG) is not designed to compress bitmaps as most other image formats. SVG is based on the Extensible Markup Language (XML) and an SVG image is in its simplest form a textual description of geometrical objects and operations. An example SVG image is shown in Table 3.1. SVG is standardised by W3C and the first specification was published in 2001.

Table 3.1 Example SVG image source and presentation.

```
<?xml version="1.0"?>
 <svg xmlns="http://www.w3.org/2000/svg">
   <g style="fill-opacity:0.7; stroke:black; stroke-width:0.1cm;">
     <circle cx="6cm" cy="2cm" r="100" style="fill:red;"
                               transform="translate(0,50)"/>
     <circle cx="6cm" cy="2cm" r="100" style="fill:blue;"
                               transform="translate(70,150)"/>
     <circle cx="6cm" cy="2cm" r="100" style="fill:green;"
                               transform="translate(-70,150)"/>
   </g>
 </svg>
```



3.2.1.5 Other formats

JPEG2000 is an advanced and complex successor of JPEG [Chr00]. It is based on wavelet transforms and promises much better compression ratios and quality than JPEG. JPEG2000 standard was approved in 2000 but it is not yet widely used. Windows Bitmap (BMP) is a format designed by Microsoft that is used to store, usually uncompressed, bitmap images.

3.2.2 Audio formats

Audio formats can be split into two categories. Some formats are designed for natural sounds and some for synthetic sounds. The formats for natural sounds are designed to compress sampled speech or music. They are usually based on a psychoacoustic model of the human hearing system. That is, they discard information from the uncompressed sampled audio signal to produce a compressed signal that sounds the same or similar to a human.

The quality of an audio format is often compared to the uncompressed audio used on CDs. Data rate of a CD is about 1400 kbit/s while speech is usually compressed using 10 kbit/s and music using 100 kbit/s. The audio formats designed for synthetic sounds use much smaller data rates.

3.2.2.1 AMR

Adaptive Multi-Rate (AMR) is a mandatory speech codec for UMTS specified by 3GPP. PSTN, all GSM speech codecs, and AMR limit the used frequency range to 300-3400 Hz. AMR uses a variable data rate to compress this 3.1 kHz speech signal. The eight possible data rates range from 4.75 kbit/s to 12.2 kbit/s. 3GPP has also specified a AMR Wideband speech codec that uses a frequency range of 50-7000 Hz and nine data rates from 6.60 kbit/s to 23.85 kbit/s. AMR is also used in some GSM networks.

3.2.2.2 MPEG-4 AAC

MPEG-4 Advanced Audio Coding (AAC) is standardised by ISO. It is a complex system offering a wide range of parameters but it is mainly used to compress music. It is newer and promises to produce better sound quality using smaller data rates than the popular MP3 format. Various companies hold patents for MPEG-4 AAC and its licence fees are based on the number of sold codec channels.

3.2.2.3 SP-MIDI

Musical Instrument Digital Interface (MIDI) is a twenty-year old standard that is not designed to compress sampled music. MIDI files are descriptions on how to produce wanted sounds and music synthetically. Scalable Polyphony MIDI (SP-MIDI) is a newer standard specifying how the same MIDI files can be played on different devices capable of playing different amount of sounds simultaneously. The MIDI Manufacturers Association specifies both of these standards. On mobile telephones SP-MIDI is used mainly for ring tones.

3.2.2.4 Other formats

Pulse Code Modulation (PCM) is used to code speech in PSTN. It is specified by ITU in G.711. The most common audio format to compress music is MPEG-1 Layer-3, which is often called MP3 [Lam01]. The open source community has developed a patent-free audio format called Ogg Vorbis as an alternative to MP3 and other formats that are covered with patents and require licensing fees [Mof01].

3.2.3 Video formats

Video is basically a sequence of images. Showing images so rapidly that the human visual system cannot anymore recognise the discrete images creates the illusion of a continuous video. Typically 25-30 images per second is enough. Video formats use the same principles to compress images as the image formats described in Section 3.2.1. This is often not enough as the data rates of sequences of compressed images are too high for transmission or storage and therefore most video formats utilise also temporal redundancy in addition to spatial redundancy. Consecutive images in the image sequence are almost identical and temporal prediction is used to reduce this redundancy.

The data rates of the video formats are one or two orders of magnitude higher than the data rates of the audio formats. A typical data rate for digital television is 4000 kbit/s and twice that much for DVD. Data rates as low as 64 kbit/s are used for video calls in fixed and wireless networks.

3.2.3.1 H.263

H.263 is the successor of H.261 specified by ITU. H.261 is designed to be used for video calls in fixed telephone networks. It supports data rates of multiples of 64 kbit/s. H.263 standard was approved in 1996. It includes several enhancements and is designed to compress video using data rates below 64 kbit/s.

3.2.3.2 MPEG-4

Moving Picture Experts Group (MPEG) video and audio formats are standardised by ISO. MPEG-4 was approved in 1999. It is the latest and most advanced video format by ISO supporting a wide range of parameters and data rates. It defines profiles and levels that are subsets of the features. The simplest profile of MPEG-4 is compatible with the baseline of H.263. MPEG-4, unlike H.263, requires similar licensing fees as MPEG-4 AAC.

3.2.3.3 Other formats

MPEG-1, approved in 1993, is designed to compress video using data rates around 1500 kbit/s. It is used on Video CDs to offer quality similar to VHS. MPEG-2 was approved a year later and it is used to compress higher quality video using higher data rates. It is used on DVDs and digital television. Many companies have developed proprietary video formats. Three widely used formats are: Apple’s QuickTime,

Microsoft's Windows Media Format, and RealNetwork's RealMedia. Many companies have also developed incompatible variants of Motion JPEG (MJPEG). It is not standardised by ISO. MJPEG does not use temporal prediction and an MJPEG file is basically a sequence of JPEG images. Many video cameras use MJPEG. Motion JPEG 2000 is specified by ISO and the first version of the standard was approved in 2002.

3.3 Media adaptation

In this thesis media adaptation means changing or modifying a media element to meet the requirements of a communicating party. Some requirements and cases when adaptation is needed were listed in Section 2.5. This section discusses how media elements can be changed, adapted, to meet the requirements.

Media adaptations can be classified into three categories with the help of the media type and media format classifications introduced in Section 3.2. The most drastic adaptation changes the media type of the adapted media element. Video can be adapted to image or slideshow, for example. Less drastic adaptation changes the media format within a media type. A GIF image can be adapted to a JPEG image or PCM audio can be adapted to AMR audio. These adaptations are often called format conversions. The third kind of adaptation changes the attributes of the media element within a media format. The resolution of a JPEG image or the frame rate of H.263 video can be decreased, for example.

Adaptation is usually done from a more complex media type or format to a simpler one. Video is considered being the most complex media type and text the simplest. This is not always the case and a common counterexample is text-to-speech adaptation that is used, for example, by the blind to interpret text. Information is lost in most cases during the adaptation, which makes the operation irreversible. The amount of information is never increased during the adaptation.

All media formats have attributes that adaptation changes and that ultimately decide whether a media element meets the requirements of a communicating party. A common attribute to all media formats is the size of the media element measured in bytes. Similar attribute for audio and video formats is data rate measured in bytes or bits per second. The maximum size of the media elements used in MMS messages is not limited but [OMA conf] specifies that the minimum supported message size is 30 kilobytes. The

size of a media element can be decreased by changing other attributes. Typical attributes are frame rate and resolution for video formats, sampling frequency and sampling precision for audio formats, resolution and number of colours for image formats. Most compressed media formats have also a compression rate attribute.

A closely related concept to media adaptation is service adaptation. When an MMS message is sent to a device that does not support MMS, the message must still be delivered somehow. A common solution is to store the MMS message to WWW and send the URI in an SMS message to the device not supporting MMS. This is often called legacy support. Service adaptation is also needed when an MMS message is sent to an e-mail address or vice versa. Service adaptation requires often adaptation of media elements.

When media adaptation is done in a network between the sender and the receiver, the network element doing the adaptation must know the hardware and software capabilities of the receiving device. There are many ways to find out the capabilities. One solution is to use WSP/HTTP headers like User-Agent and Accept [RFC 2616] when an MMS client fetches the MMS message from an MMSC. A drawback of this solution is the need of a database containing the capabilities of different user agents. A more complex solution is to use the User Agent Profiles method described in [OMA UAPr]. Both of these solutions have the limitation that the adapting network element finds out the capabilities only after the MMS client initiates the fetching. There exists also other solutions but no solution has been established as the ultimate solution and the search continues.

Adapting an MMS message, a composite of media elements, to match the capabilities of the receiving device is a very complex optimisation problem. Adaptation should use as few computational resources as possible but still produce the best feasible quality. Evaluating the quality of the adapted media elements can often be done only subjectively and there is no simple algorithm to decide what adaptation operations to perform on which media elements of an MMS message. Adapting MMS messages is out of scope of this thesis and the focus is on performing given adaptation operations on one media element at a time.

4 Image coding and JPEG

This chapter gives an overview of image coding using JPEG as an example of an image format. The first section describes general image coding principles like colour spaces and compression techniques. Common terms and concepts are defined, explained, and illustrated. The second section takes a closer look at discrete cosine transform (DCT) that is used in JPEG and in many other media formats. The third section describes Huffman coding that is also used in JPEG and other media formats. The fourth section introduces the JPEG image format and contains a detailed example of JPEG encoding and decoding.

4.1 Basics of image coding

In this chapter, image means digital, two-dimensional, rectangular array of pixels that is interpreted by a human visual system or by a computer [Gon01], [Shi99]. A pixel, picture element, is a rectangle, often a square, which is coloured with one colour. Hundreds of thousands or even millions of pixels put together create the illusion of an image. The world a human sense is analogue but all modern computers are digital. Digital images can be created with computers or by digitising analogue images with scanners or digital cameras, for example. Computer monitors and other displays or printers can be used to convert digital images to analogue images for viewing.

An example image, the famous Lena [Hut01], is shown in Figure 4.1. The size of the Lena image, array of pixels, shown in the left side of the figure is 512 pixels by 512 pixels. This is called the resolution of the image. A small portion of the image, 32 by 32 pixels, is magnified by seven times and shown in the right side of the figure. The individual pixels can be seen in the enlargement.

An uncompressed image is usually stored to a file as an array of values describing the colours of the corresponding pixels. The example image uses 24 bits to describe the colour of a pixel. This means that over 16 million colours can be expressed, often called true colours. Different colour spaces are described in detail in Section 4.1.1. The size of the Lena image measured in pixels is 262 144 and the size of the file is 786 432 octets plus the requirements of the image format. Different image formats were introduced in Section 3.2.1.

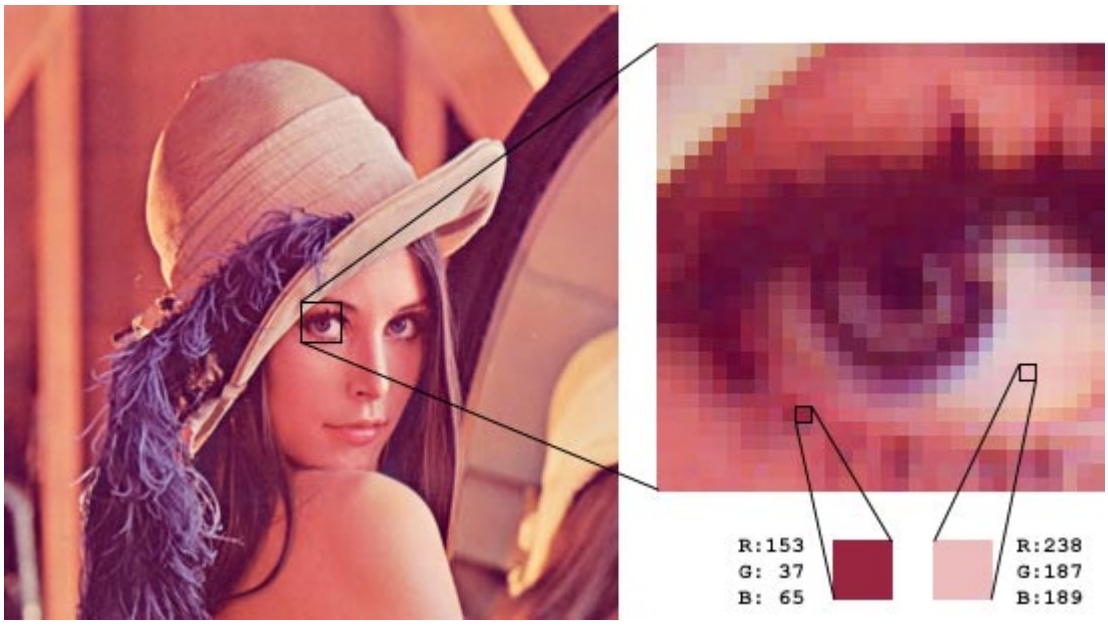


Figure 4.1 Example image.

The resolutions of images vary greatly depending on the usage of the images. The sizes of the icons used in graphical user interfaces or the thumbnail images used in the WWW are around 1000 pixels. The size of most images in the WWW is something between 10 000 and 100 000 pixels. The sizes of digital photographs and professional images can be anything from 100 000 pixels up to millions of pixels.

The simplest images have only two colours, often black and white. Greyscale images have usually up to 256 colours and colour images up to billions. An image is said to tell over a thousand words and in most cases storing or transmitting an image definitely requires more bytes than storing or transmitting a thousand words of text. Different image compression techniques are introduced in Section 4.1.2. Advanced compression techniques like fractals and wavelets are out of the scope of this thesis.

4.1.1 Colour spaces

The colour of an object in the real world depends on the wavelength of the light it reflects. When a digital image is shown on a computer monitor or on other display, the display emits light that tries to behave like the light reflected from the objects in the real world. Different methods are used to emit the light but most of the techniques construct the emitted light from three components: red, green, and blue. These three colours, RGB, are often used to describe the colours of digital images and video but also other colour spaces, colour systems, exist and are needed.

RGB is a somewhat intuitive, additive colour space. Almost all the different colours a human can sense can be constructed by adding different amounts of red, green, and blue. The amount of a colour component is usually expressed by using 8 bits, or values from 0 to 255. When no colour components are present (R=0, G=0, B=0), the colour is black and when all the components with the maximum amounts are present (R=255, G=255, B=255), the colour is white. The RGB values of two pixels are shown in Figure 4.1. The coordinates of the left pixel in the original Lena image are x=258 and y=274 (origin is in the top left corner). It is dark red as can be seen from the high value of the red component and the small values of the other components. The right pixel (x=274, y=271) is light red as is indicated by the higher values of all components. There are no other pixels in the Lena image with the same RGB values as these two pixels.

Another widely used colour space is called YC_BC_R. It is very similar to YUV and YIQ, which are used in TV colour systems PAL and NTSC, respectively. YC_BC_R is also based on three components like RGB. Y is a luminance component while C_B and C_R are chrominance components. The luminance component, the only component in greyscale images, contains the brightness information of the image and the chrominance components contain the colour information. The conversion between RGB and YC_BC_R can be done using the formulas shown in Table 4.1.

Table 4.1 Formulas for colour conversions between RGB and YC_BC_R. [JFIF]

$Y = 0.299*R + 0.587*G + 0.114*B$ $C_B = -0.169*R - 0.331*G + 0.500*B + 128$ $C_R = 0.500*R - 0.419*G - 0.081*B + 128$	$R = Y + 1.402*(C_R - 128)$ $G = Y - 0.344*(C_B - 128) - 0.714*(C_R - 128)$ $B = Y + 1.772*(C_B - 128)$
(a) From RGB to YC_BC_R	(b) From YC_BC_R to RGB

4.1.2 Compression

Uncompressed images require often too many bytes for storage or transmission, as was seen earlier in this chapter. Different compression techniques are used to solve this problem. General, lossless, compression techniques like LZW can be used to compress images but in many cases they are not enough. The general compression techniques do not utilise the special characteristics of images.

Lossless compression means that no information is lost during the compression and that the decompressed image is exactly, bit by bit, the same as the original image. Lossy

compression means that information is deliberately lost during the compression and that the decompressed image differs from the original image. The compression ratios of the lossless image compression techniques can be an order of magnitude lower than the compression ratios of the lossy image compression techniques providing about the same image quality.

A common goal for all image compression techniques is preserving the quality of the original image as well as possible. For lossless techniques this is not a problem, as they do not alter the original image. Lossy compression usually decreases the quality but using suitable compression parameters and rates, a human cannot tell a lossy compressed image from the original. Examples of lossy compressed JPEG images are shown in Figure 4.2. The compression ratio of the Lena image in the left is 20:1 and the quality is only slightly decreased. The compression ratio of the Lena image in the right is 50:1 and the quality is considerably decreased. The same regions have been magnified as in Figure 4.1 to illustrate the effects of the compression, the compression artefacts. Usually compression artefacts are not wanted but some artists use them and other digital artefacts in their work.

Lossy image compression techniques are based on a psychovisual model of the human visual system in a similar way as lossy audio compression techniques are based on a psychoacoustic model of the human hearing system. Special care must be taken when a computer is used to interpret a lossy compressed image because the human visual system differs a lot from a computer.

One of the reasons why the YC_BC_R colour space is used is the knowledge that the human visual systems is much more sensitive to the changes of luminance than chrominance. A simple way to compress images is to subsample the chrominance components. Subsampling means reducing the resolution. If the resolution of the chrominance components is halved horizontally, the format is called 4:2:2. If the resolution is also halved vertically, as is usually done in JPEG, the format is called 4:2:0. If no subsampling is performed, the format is called 4:4:4.

One characteristic of photographs and other continuous-tone images is the correlation between adjacent pixels. As can be seen from Figure 4.1, the colour of a pixel is in most cases very close to the colour of the pixel next to it. This is called spatial redundancy.

Image compression techniques based on mathematical transforms are used to reduce the spatial redundancy. Transforms are used in mathematics to transform problems from one domain to another because certain problems are easier to solve in certain domains. Transforms are used in image coding to transform images from a pixel domain to a transform domain. The pixel domain, sometimes called spatial domain, means the array of pixels seen earlier in this chapter. The transform domain, or frequency domain, is just another representation of the same information. The transform itself is a lossless operation but it is done because the redundant information is easier to find and lose in the transform domain than in the pixel domain.

Figure 4.2 Two JPEG images with compression ratios of 20:1 and 50:1.

4.2 Discrete cosine transform

Discrete cosine transform (DCT) is the most widely used transform in image and video coding. It was published in the 1970s. DCT is used in JPEG, H.263, and MPEG video formats, for example. Two-dimensional DCT for 8x8 pixels as used in JPEG is given in Table 4.2. Forward DCT takes 64 pixel values s_{yx} and gives 64 DCT coefficients S_{vu}.

The DCT coefficient in the top left corner of the coefficient matrix, S_{00}, is called DC component and the other coefficients are called AC components. An example of DCT calculation is shown in Section 4.4.2. In real applications, DCT is never calculated using the formulas given here but fast algorithms introduced in Section 5.3.

Table 4.2 Formulas for Forward DCT and Inverse DCT. [JPEG]

FDCT	$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$	$C_u, C_v = \frac{1}{\sqrt{2}}$ for $u, v = 0$ $C_u, C_v = 1$ otherwise
IDCT	$s_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{vu} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$	

Forward DCT can be thought of transforming a block of 8x8 pixels to a sum of weighted cosine signals, whose weights are expressed by the DCT coefficient matrix. The coefficients near the top left corner represent the low spatial frequencies and the coefficients near the bottom right corner represent the high spatial frequencies. The low frequencies represent slow, gradual changes and the high frequencies represent fast, sharp changes in the pixel domain. Because of the spatial redundancy, the low frequencies dominate the high frequencies and the forward DCT concentrates the information, or energy, contained in the 8x8 block to the top left portion of the coefficient matrix. This means that the coefficients near the DC coefficient differ a lot from zero while the other coefficients are close to zero.

DCT is a lossless operation as was said earlier. Information is lost in transform based compression techniques by quantising the DCT coefficients. Quantisation means dividing each element of the coefficient matrix by the corresponding element of the quantisation matrix and rounding the result to the nearest integer. An example is shown in Section 4.4.2. The coefficients in the bottom right portion of the matrix are quantised more coarsely than the other coefficients. The goal is to round as many of the high frequency coefficients to zero as possible. The chrominance components are also quantised more coarsely than the luminance component. In both cases the coarser quantisation utilises the characteristics of the human visual system. In the transform domain, information that is less sensitive to a human can be separated from the more sensitive information and the amount of it can be reduced as much as is wanted.

4.3 Huffman coding

Text written in natural languages is constructed using a finite alphabet. Each character of the alphabet is usually coded using a fixed number of bits, seven in ASCII. Different characters appear in text with different frequencies, or probabilities. Coding characters, or any symbols, with different occurrence probabilities using a fixed number of bits introduces redundancy. Entropy is a basic concept of information theory meaning the amount of information in a piece of data measured in bits. Entropy coding techniques try to reduce the redundancy and decrease the size of the data closer to the entropy of it. Lossless techniques cannot compress data to a smaller space than its entropy.

Huffman coding is the most widely used entropy coding technique. It was published in the 1950s. Huffman encoder builds a table that is indexed by the fixed length source symbols and contains a variable length codeword for each symbol. The codewords for the more frequent symbols are shorter than the codewords for the less frequent symbols. No codeword is a prefix of another codeword. Source data is encoded to a bit stream by replacing each source symbol with the corresponding codeword. Huffman decoder decodes the compressed bit stream using the same table the encoder constructed.

Huffman encoding of the ASCII string “mmsmessage” is shown in Table 4.3. The first column of the first row shows the five source symbols and their number of occurrences. In each step of the encoding, two symbols with the least occurrences are combined to a new symbol with the sum of the number of the occurrences. The second row shows the codewords in each step of the encoding. In each step, zero is prefixed to the codeword of the symbol with the least occurrences and one is prefixed to the other symbol. The last column of the second row shows the final Huffman table. The encoded bit stream is 000001001001011101110 and the compression ratio is 3.2.

Table 4.3 Huffman encoding of the ASCII string “mmsmessage”.

m 3 s 3 e 2 **a** 1 **g** 1	m 3 s 3 **e** 2 **ag** 2	**m** 3 **s** 3 eag 4	**ms** 6 **eag** 4	mseag 10
m s e a g	m s e a **0** g **1**	m s e **0** a **1**0 g **1**1	m **0** s **1** e 0 a 10 g 11	m **0**0 s **0**1 e **1**0 a **1**10 g **1**11

4.4 JPEG image format

4.4.1 Overview

ISO and the International Electrotechnical Commission (IEC) formed a working group to develop a standard for encoding of continuous -tone greyscale and colour images in 1986. The group developed the standard in collaboration with ITU and it was published as T.81 by ITU in 1992 and as 10918-1 by ISO/IEC in 1994. The standard does not explicitly name the image format but it is known as Joint Photographic Experts Group (JPEG) after the informal name of the committee that created it [JPEG], [Wal92].

The JPEG standard aims to be applicable to a broad range of applications. It specifies processes for encoding and decoding images as well as the interchange format, which is a compressed image data representation including all table specifications used in the encoding process and needed in the decoding process. The interchange format is needed to be able to exchange JPEG images between applications but it is not enough. The JPEG standard does not specify any colour space or the aspect ratio of the pixels, for example. These application-dependent issues are specified in JPEG file formats. The most widely used file format is called JPEG File Interchange Format (JFIF), which was published in 1992 [JFIF]. This minimal and restrictive ad hoc file format has been established as the de facto standard. The official file format for JPEG called Still Picture Interchange File Format was published by ISO/IEC and ITU in 1996. That was too late and the file format was dismissed.

The JPEG standard defines four modes of operation: sequential DCT-based, progressive DCT-based, lossless, and hierarchical. The DCT-based encoding process groups input component's samples into 8x8 blocks and encodes them in three steps: forward DCT, quantisation, and entropy coding. A detailed example is shown in the next section. In the sequential mode, blocks are coded from left to right and from top to bottom. In the progressive mode, blocks are coded in different order to make it possible for the decoder to present the image progressively. The lossless mode is not based on DCT but on prediction and entropy coding. In the hierarchical mode, the image is coded at multiple resolutions to enable a progressive presentation. The lossless and hierarchical modes are usually not supported by applications and virtually never used. In addition to the Huffman coding, the JPEG standard specifies also another entropy coding technique: arithmetic coding. It is patented, never used, and not discussed in this thesis.

4.4.2 Encoding and decoding example

The JPEG standard defines many different parameters and modes of operation, as was seen in the previous section. Applications are not required to support all the features and to ensure interoperability between the applications, a baseline coding process is defined. It is sufficient for most applications and users. The mode of the baseline process is the sequential DCT-based and it uses Huffman coding as the entropy coding technique.

This section shows an example of the baseline coding. The last 8x8 block of the Lena image is encoded and decoded. The RGB components (Figure 4.3) and the luminance component calculated using the formulas from Table 4.1 are shown in Table 4.4.

Table 4.4 RGB and luminance components of the last 8x8 block of the Lena image.

107 102 110 127 124 133 146 153 108 115 129 133 141 145 156 157 113 122 135 137 152 158 161 161 120 129 141 147 160 165 162 162 128 135 147 154 170 172 166 167 132 142 151 162 174 173 172 177 147 152 162 173 177 179 181 185 147 152 162 173 177 179 181 185	30 24 35 43 44 41 45 55 34 33 42 51 53 53 61 64 30 38 45 49 71 60 55 64 41 52 56 62 59 61 63 56 48 55 60 64 68 71 67 63 53 51 65 67 66 73 68 62 69 62 68 74 66 70 71 74 69 62 68 74 66 70 71 74	61 57 64 72 63 65 68 73 59 63 73 70 76 65 72 80 58 67 68 69 84 69 69 81 64 78 71 79 70 72 77 72 73 74 82 83 82 82 77 82 76 74 80 85 79 84 76 79 88 85 80 89 77 79 81 81 88 85 80 89 77 79 81 81	57 51 61 71 70 71 78 86 59 61 72 78 82 82 91 94 58 66 75 78 97 90 88 95 67 78 83 89 90 93 94 90 75 81 89 93 100 102 98 96 79 81 92 97 100 104 100 98 94 92 97 105 100 104 105 108 94 92 97 105 100 104 105 108
(a) Red	(b) Green	(c) Blue	(d) Luminance

Using the formula from Table 4.2, forward DCT is calculated for the luminance component (d) after 128 is subtracted from each element, or sample, of the component. The result, DCT coefficient matrix, is shown in Table 4.5 (a). The JPEG standard does not require the use of any specific quantisation table but example tables are given for luminance (c) and chrominance. Matrix (a) is quantised using table (c) after each element of the table is divided by 2. The resulting coefficients are shown in matrix (b).

Table 4.5 Forward DCT, quantisation, coefficient ordering, and inverse DCT.

-325.2 -65.4 -19.6 -6.0 4.5 2.1 2.4 0.4 -84.4 -18.4 0.7 -3.2 3.1 -3.0 -1.4 5.3 -9.3 1.7 13.5 -4.2 8.8 7.2 1.3 -2.2 -8.9 1.8 0.8 1.1 0.6 5.1 2.5 -1.4 -6.7 5.8 1.8 -0.3 -1.0 2.8 -1.0 -3.2 -4.2 4.5 2.0 -0.9 1.3 0.7 -0.8 -2.3 -8.8 -3.4 -4.5 2.7 1.7 -4.3 2.2 2.3 2.7 -1.7 -0.2 2.1 3.5 -4.7 1.4 2.1	-41 -11 -4 -1 0 0 0 0 -14 -3 0 0 0 0 0 0 -1 0 2 0 0 0 0 0 -1 0 0 0 0 0 0 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	16 11 10 16 24 40 51 61 12 12 14 19 26 58 60 55 14 13 16 24 40 57 69 56 14 17 22 29 51 87 80 62 18 22 37 56 68 109 103 77 24 35 55 64 81 104 113 92 49 64 78 87 103 121 120 101 72 92 95 98 112 100 103 99
(a) Luminance DCT coefficients	(b) Quantised coeffs	(c) Quantisation table
-41 -11 -14 -1 -3 -4 -1 0 0 -1 -1 0 2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 1 5 6 14 15 27 28 2 4 7 13 16 26 29 42 3 8 12 17 25 30 41 43 9 11 18 24 31 40 44 53 10 19 23 32 39 45 52 54 20 22 33 38 46 51 55 60 21 34 37 47 50 56 59 61 35 36 48 49 57 58 62 63	55 59 63 67 71 75 80 83 57 62 69 75 80 84 88 91 61 67 77 84 88 90 92 93 68 74 84 90 93 91 90 89 74 80 89 95 97 95 93 91 80 86 93 99 101 102 102 102 88 92 97 101 103 104 106 108 95 97 99 100 100 101 104 106
(d) Zig-zag ordered coefficients	(e) Zig-zag matrix	(f) Luminance

Using the formula from Table 4.2, inverse DCT is calculated for the quantised DCT coefficients in matrix (b). The result is the luminance component shown in matrix (f). This can be compared to the original luminance component in Table 4.4. To illustrate the effects of DCT, the 8x8 RGB block is shown before and after coding in Figure 4.3.

Figure 4.3 The last 8x8 block of the Lena image before and after JPEG coding.

As can be seen from matrix (b) in Table 4.5, most of the coefficients in the bottom right portion of the matrix are zero after quantisation. In preparation for the Huffman coding, the DCT coefficients are run-length coded. First, matrix (b) is ordered to zig-zag order using matrix (e). The goal is to make as long sequences of zeros as possible. The result is shown in (d). Entropy coding and the coded bit stream are shown in Table 4.6.

First, all coefficients in (d) are transformed to intermediate symbols, which are then Huffman encoded using separate tables for DC and AC components. The JPEG standard does not require the use of any specific Huffman table but example tables are given for luminance and chrominance. The intermediate symbol format is (size)(amplitude) for DC and (run-length, size)(amplitude) for AC. Size means the number of bits used to represent the amplitude and run-length means the number of consecutive zeros. The first part of the intermediate symbol is encoded using a Huffman table and it is appended by the binary representation of the second part of the intermediate symbol, the amplitude.

Table 4.6 Run-length and Huffman encoding of the quantised DCT coefficients.

The difference of two consecutive DC coefficients is coded instead of the DC. DIFF = DC – PREVIOUS_DC = -41 – (-70) = 29 29 -> (5)(29) -> (110)(11101)	-11 -> (0,4)(-11) -> (1011)(0100) -14 -> (0,4)(-14) -> (1011)(0001) -1 -> (0,1)(-1) -> (00)(0) -3 -> (0,2)(-3) -> (01)(00) -4 -> (0,3)(-4) -> (100)(011) -1 -> (0,1)(-1) -> (00)(0) -1 -> (2,1)(-1) -> (11100)(0) -1 -> (0,1)(-1) -> (00)(0) 2 -> (1,2)(2) -> (11011)(10) 1 -> (6,1)(1) -> (1111011)(1) (0,0) -> 1010
(a) DC coefficient	(b) AC coefficients
110 11101 1011 0100 1011 0001 00 0 01 00 100 011 00 0 11100 0 00 0 11011 10 1111011 1 1010	
(c) JPEG encoded luminance component (68 bits, the compression ratio is 7.5:1).	

5 JPEG adapter

This chapter describes the implementation part of this thesis. The first section introduces the features of the implemented program, JPEG adapter, and gives reasons why it was implemented. Digital signal processors in general and the used digital signal processor in particular are described in the second section. The third section shows how discrete cosine transform is calculated in real applications. The implementation of JPEG adapter is described in detail in the fourth section. Reused software components like the operating system and JPEG library are also introduced.

5.1 Features and reasoning

JPEG adapter is a program that runs on a digital signal processor (DSP) and adapts JPEG images. It follows the client-server, or request-response, computing model. JPEG adapter is a server and it waits for adaptation requests. A client initiates the adaptation by sending an adaptation request to JPEG adapter and if the response is positive, the client sends also the image to be adapted. JPEG adapter adapts the image and sends it back to the client. Signalling and image transport use proprietary protocols running on top of User Datagram Protocol (UDP).

An adaptation request contains a target resolution and a target file size for the image to be adapted. JPEG adapter tries to adapt the image to fulfil these restrictions. Only operation it can perform is to horizontally and vertically halve the resolution of the image. Halving the resolution of a JPEG image is not as trivial as it may first sound. Resolution is defined by means of pixels and the resolution is halved in the pixel domain. A JPEG image is transformed to the pixel domain by applying entropy decoding, dequantisation, and inverse DCT. After the resolution is halved in the pixel domain, forward DCT, quantisation, and entropy encoding are applied. Methods that halve the resolution of a JPEG image in the transform domain exist but they are out of the scope of this thesis.

JPEG adapter is capable of adapting baseline JPEG images, whose resolution is smaller than 512x512 pixels and file size is less than 100 000 octets. A client running on a PC is also implemented to be able to test JPEG adapter. It is described along the test environment in Section 6.1.

The purpose of the implementation part of this thesis is to study closer the techniques and algorithms introduced in the previous chapters and to gather knowledge on running them on a DSP. Issues related to differences between DSPs and general-purpose processors as well as differences between image coding and audio coding are expected to arise during the implementation. Such issues will be discussed. The aim is not to fully optimise the program but find out the algorithms that may need optimisation and discuss how they could be optimised. The aim is not to design and implement a commercial product or anything that is useful as such but rather explore the unknown.

Indeed, JPEG adapter is not that useful as such. Halving the resolution of JPEG images of MMS messages is too coarse for real applications, for example. Resolution halving is chosen to be implemented because it uses the most important algorithms of image and video coding: DCT, quantisation, and Huffman coding. Reducing the resolution arbitrarily would be more useful for real applications but halving is easier to implement and it fulfils the requirements. Instead of just implementing and testing the separate algorithms with artificial test data, it is seen useful to adapt real JPEG images and to experiment with somewhat realistic application.

One reason why JPEG adapter is kept simple is to be able to easily run it on different DSPs and general-purpose processors. It can be used to benchmark different processors for media adaptation as it implements many of the algorithms needed to adapt most image and video formats. Software reuse is one of the design principles of JPEG adapter and existing modules and components are used whenever possible. Connectionless UDP is used for image transport instead of TCP because UDP is available for a wider variety of processors than TCP.

JPEG adapter is tested using TMS320VC5510 DSP by Texas Instruments. It is described in the next section after DSPs are discussed in general. The chosen DSP is by no means optimal for image or video coding but one of the reasons why it was chosen is because it is widely used for audio coding in mobile telephones and networks. While not being optimal, the processor is definitely suitable for the purpose. It is not the latest offering from Texas Instruments and thus the developing tools for it should be mature enough to be able to concentrate on the real work. When developing software for a brand new DSP, much of the time is often spent on debugging the development tools and the processor itself.

5.2 Digital signal processors

5.2.1 Overview

There is no clear definition what is the difference between a digital signal processor (DSP) and a general-purpose processor (GPP). It can be said that any processor that is designed to execute efficiently signal processing algorithms is a DSP. GPPs and DSPs have adopted functionality from each other in recent years but there is still a clear need for the two distinct processor types. The biggest GPP manufacturer by far is Intel with its Pentium processors followed by Advanced Micro Devices. The biggest DSP manufacturer is Texas Instruments ahead of Motorola, Agere Systems (Lucent Technologies), and Analog Devices. Most GPPs are used in PCs while most DSPs are used in mobile telephones. A typical PC is 5-10 times more expensive than a typical mobile telephone and similar difference can be found in the prices of the processors. Computing power is the most important property of a GPP but power consumption is at least as important property of a DSP designed for mobile devices.

DSP architecture design is driven by signal processing algorithms [Eyr00]. They have changed little since the commercial introduction of DSPs in the early 1980s. Most signal processing algorithms like filters and transforms can be expressed by the equation $\sum x_i h_i$. That is, input data sample x_i is multiplied by coefficient h_i and the result is added to a running sum. This operation is often called multiply and accumulate (MAC). GPPs have traditionally calculated a multiplication as a series of add and shift operations taking many clock cycles. The number of multiplications is therefore tried to minimise in algorithms computed on GPPs. As multiplications are essential in signal processing, DSPs have been capable of calculating a multiplication in one clock cycle since the beginning. Most DSPs have a specific MAC unit that can calculate the whole operation in one cycle including the instruction fetch and the two data fetches.

To be able to fetch the instruction and data at the same time, they are stored in two separate memories. This is called Harvard architecture. Most DSPs use a modified version of it to enable more than one data fetches in one clock cycle. Many GPPs store program and data in the same memory. This is called von Neumann architecture after John von Neumann who is said to be the father of the modern computer, game theory, cellular automata, and undoubtedly one of the greatest minds of the previous century. To be able to calculate efficiently the repetitive MAC operations described by the above

equation, most DSPs can execute loops without any overhead. No cycles are spent on updating or testing the loop counter or branching from the end of the loop to the start.

Most modern GPPs support both fixed-point and floating-point data representations while most DSPs support only the former. Fixed-point operations are simpler and consume less power than floating-point operations, but they are less precise. Other design choices that separate DSPs from GPPs are specialised instruction sets and addressing modes. These features make it possible to utilise the underlying specialised hardware but they make the writing of an efficient compiler more difficult. Therefore, programs for DSPs are usually written in assembly langue or C rather than in higher-level languages like C++ or Java. Algorithms are often written in assembly language and control code in C.

The computing power of processors is traditionally measured using the unit Million Instructions Per Second (MIPS). MIPS figures are theoretical and have little relevance to the real performance of processors as is seen later in this section. More useful figures can be achieved by measuring real applications but since it is difficult to port them to different processors, a set of algorithms is used instead. The algorithms are chosen to represent the computational tasks the processors are usually used for. Two companies have established de facto standards for processor benchmarks: Standard Performance Evaluation Corporation for GPPs and Berkeley Design Technology for DSPs.

5.2.2 Used digital signal processor

Texas Instruments has currently three DSP processor families, or platforms: control optimised TMS320C2000, power efficient TMS320C5000, and high performance TMS320C6000. These processor platforms are further divided into device generations, or processor cores. The power efficient TMS320C5000 platform consists of two device generations: TMS320C54x and TMS320C55x. The latter generation was announced in the year 2000 and there are currently four processors in it. The first processor of the TMS320C55x generation was the TMS320VC5510 running at 160 MHz but the newer version running at 200 MHz is used in this thesis [TI datam]. In 2002, the suggested retail price of the used DSP was 25 US dollars when buying 10 000 processors.

TMS320VC5510 is a fixed-point DSP capable of executing two MAC operations in one clock cycle and therefore its computing power is said to be 400 MIPS. Its score in

BDTImark2000 benchmark is 970. For comparison, the same figures for TMS320C64x running at 720 MHz are 5760 MIPS and 6480 in BDTImark2000 benchmark. It can be seen that the MIPS figure is 14.4 times higher but the BDTImark2000 score is only 6.7 times higher. This follows from the fact that real algorithms are not able to utilise in every cycle the eight computing units TMS320C64x offers. GPPs cannot be compared directly to DSPs but the floating-point BDTImark2000 score for Intel's Pentium III at 1400 MHz is 3130. TMS320VC5510 consumes power about 0.3 W while the Pentium consumes about 30 W. Pentium 4 running at 1400 MHz consumes power about 55 W.

In addition to the two 17-bit x 17-bit MAC units, TMS320VC5510 has one 40-bit and one 16-bit arithmetic logic unit, four 40-bit accumulators, and one 40-bit barrel shifter. On-chip peripherals include two 20-bit timers, a Direct Memory Access controller, and three Multichannel Buffered Serial Ports. Instructions are variable length ranging from 8 bits to 48 bits and a 192-kilobit instruction cache can be used. Core supply voltage is 1.6 V and I/O voltage is 3.3 V.

From the programmer's point of view, the biggest differences between Pentiums and TMS320VC5510 are the size of a byte, the order of bytes within a word, and the size of the memory. Byte is often thought to mean eight bits and therefore type `char` in C programming language is thought to mean eight bits as well. This is a misconception as [ISOC] says "a byte is composed of a contiguous sequence of bits, the number of which is implementation-defined" and "it is possible to express the address of each individual byte of an object uniquely". The minimum size of a byte is defined to be eight bits. The smallest addressable data unit in TMS320VC5510 is 16 bits and therefore the size of a byte and type `char` is 16 bits. Octet means always eight bits and it should be used when confusion may occur. The leftmost byte within a word is the most significant and the rightmost byte is the least significant in TMS320VC5510 while the opposite order is used in Pentium. The former order is called big-endian and the latter little-endian.

TMS320VC5510 is based on a modified Harvard architecture with one program bus, three data read buses, two data write buses, and six address buses. The size of the memory space is 8 megabytes. There are three different types of internal on-chip memory: 32 kilobytes of dual-access RAM, 128 kilobytes of single-access RAM, and 16 kilobytes of ROM. The rest of the memory space can be used as external memory. It is worth remembering that here a byte means 16 bits.

5.3 Computing discrete cosine transform

As was said earlier, real applications do not use the formulas from Table 4.2 to calculate DCT. It can be easily seen that the complexity of the two-dimensional forward DCT for NxN pixels in Table 4.2 is $O(N^4)$. The formula for forward DCT can be rewritten as in Table 5.1 to illustrate the separability of 2-D DCT. This property means that 2-D DCT can be calculated by first calculating 1-D DCT for each row and after that 1-D DCT for the resulting columns. The complexity of 1-D forward DCT in Table 5.1 is $O(N^2)$ and the complexity of 2-D DCT is reduced to $O(N^3)$. It is possible to calculate 1-D DCT in $O(N \log N)$ using the same ideas that are used in fast fourier transform (FFT) to calculate discrete cosine transform. This reduces the complexity of 2-D DCT to $O(N^2 \log N)$. FFT was invented in the 1960s and it is often considered being one of the most important algorithms of computational science and engineering [Roc00].

[Loe89] describes how an 8-point DCT can be calculated using only 11 multiplications and 29 additions. This algorithm or variations of it are used in many applications including the JPEG library introduced in the next section.

TMS320VC5510 contains hardware extensions for image and video applications [TI hwexs]. There are extensions for DCT, motion estimation, and pixel interpolation. Hardware extensions for DCT mean 30 special instructions that can be used to calculate forward and inverse DCTs for 4x4 and 8x8 pixels. A function library containing C-callable functions utilising the hardware extensions is available [TI lib]. Forward DCT for 8x8 pixels takes 240 cycles and inverse DCT takes 168 cycles. The library contains also the same functions without the hardware extensions. They take 1082 and 676 cycles, respectively. The use of the library is discussed in Section 6.4.3.

Table 5.1 Formula for forward DCT rewritten to illustrate separability.

2-D FDCT	$S_{vu} = \frac{1}{2} C_v \sum_{y=0}^{7} \cos\frac{(2y+1)v\pi}{16} \left[\frac{1}{2} C_u \sum_{x=0}^{7} s_{yx} \cos\frac{(2x+1)u\pi}{16} \right]$	C_u and C_v are defined as in Table 4.2.
1-D FDCT	$S_u = \frac{1}{2} C_u \sum_{x=0}^{7} s_x \cos\frac{(2x+1)u\pi}{16}$	

5.4 Implementation

5.4.1 Overview

JPEG adapter is implemented in C programming language. Existing operating system, IP stack, and JPEG library are reused. They are all introduced later in this section. The most essential part of JPEG adapter is the application process, which is described using pseudocode in Figure 5.1. The message sequence chart in the same figure shows the control message flow between a client and JPEG adapter during a successful adaptation.

```
PROCESS(JPEG_adapter) {
    set state to WAITING_AREQ;

    while (TRUE) {
        receive_UDP_packet();

        if (UDP_packet is control) {
            process_received_ctrl_msg();

            if (state is RECEIVING_IMAGE) {
                send_ctrl_msg(ACK);
            } else if (state is SEND_IMAGE) {
                send_image();
                set state to WAITING_AREQ;
            }
        } else if (UDP_packet is data) {
            process_received_data_msg();

            if (state is IMAGE_RECEIVED) {
                adapt_received_image();
                send_ctrl_msg(ACOM);
                set state to WAITING_ACK;
            }
        }
    }
}
```

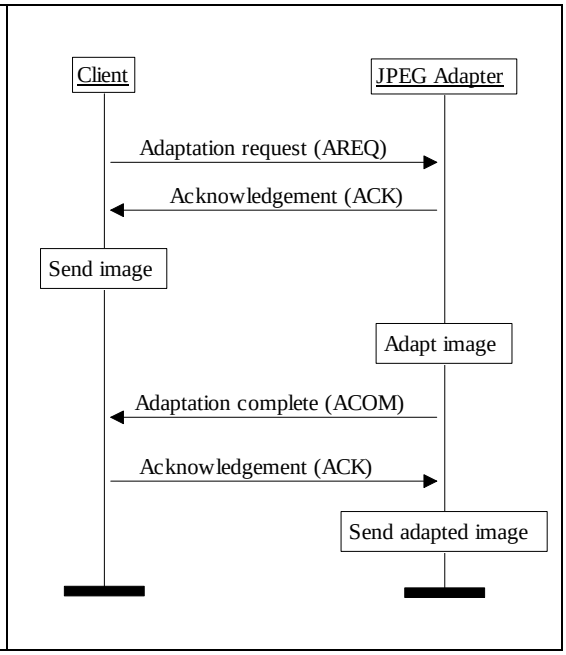


Figure 5.1 Pseudocode and message sequence chart of JPEG adapter.

The process is basically a simple state machine that processes incoming UDP packets. Received UDP packet may contain either image data or a control message. Packets are processed by different functions that also update the state of the process. The five possible states and the transitions between them are described in Table 5.2.

Table 5.2 State transitions of successful image adaptation.

State	Input	Output	Next state
WAITING_AREQ	Adaptation request	Acknowledgement	RECEIVING_IMAGE
RECEIVING_IMAGE	UDP data packets	None	IMAGE_RECEIVED
IMAGE_RECEIVED	None	Adaptation complete	WAITING_ACK
WAITING_ACK	Acknowledgement	None	SEND_IMAGE
SEND_IMAGE	None	UDP data packets	WAITING_AREQ

Adaptation request contains the file size of the image to be adapted in addition to the target values for resolution and file size. If all the values are within required limits, JPEG adapter sends a positive acknowledgement to the client, which then sends the image to JPEG adapter. Image transport over UDP is described more closely in Section 6.3.1. JPEG adapter stores the image to be adapted to the external memory of the DSP. Received image is adapted according to the request using the JPEG library. After the adaptation is done, JPEG adapter sends a control message to the client containing the file size of the adapted image. The client replies with acknowledgement and finally JPEG adapter sends the adapted image to the client.

5.4.2 Operating system

Operating system is a computer program that manages the hardware resources of a computer. These resources often include the processor, memory, and I/O devices like keyboard, printer, display, and modem. Operating system takes care of tasks like memory management, virtual memory, file system, and scheduling of processes. Operating system is typically the most complicated program run on PCs.

As systems based on DSPs are usually simpler than systems based on GPPs, the operating systems run on DSPs are also simpler. Enea Embedded Technology's OSEck operating system is used to implement JPEG adapter [OSE]. The operating system is basically a scheduler as no support for features like file system or virtual memory is needed. OSEck is a real-time operating system, which means that it is designed to run programs that react to the events of the real world in real time as they occur. DSPs are usually designed real-time operating systems in mind while GPPs are not.

OSEck's memory management is based on the concepts of buffers and pools. A buffer is a section of memory that an application may allocate using a system call. The buffers are allocated from the buffer pools that are created using a system call. Each pool has buffers of eight different sizes that are defined when the pool is created. There may be up to 16 pools. TMS320VC5510 limits the size of a pool to the size of a memory page, which is 64 kilobytes.

5.4.3 IP stack

IP stacks are usually designed to implement a large set of features and to support TCP, which make them too complicated for DSPs. UDP is more suitable for transport of

delay sensitive data than TCP. UDP as a protocol is trivial to implement compared to TCP. An IP stack that is described in [Sav01] is used to implement JPEG adapter. It is designed specially for DSPs and UDP. The application programming interface of the used IP stack is based on the BSD socket interface, which is the de facto standard of socket interfaces.

5.4.4 JPEG library

Probably the most widely used JPEG library is the one made by the Independent JPEG Group (IJG) [libjpeg]. Web browsers Mozilla, Opera, and Microsoft's Internet Explorer are all based on it. The first version of the library was released in 1991. The latest version 6b from 1998 is used to implement JPEG adapter. The library is distributed for free in the Internet and permission is granted to use, copy, modify, and distribute it for any purpose, without fee.

IJG JPEG library supports two out of the four JPEG modes of operation: sequential and progressive, while it does not support hierarchical and lossless modes. Arithmetic entropy coding is not supported for legal reasons. In addition to the support of the JPEG standard, the library supports JFIF and many other features that applications processing JPEG images need like colour space conversions and subsampling. The library comes with three example applications including applications for compressing and decompressing JPEG images.

IJG JPEG library is written in C programming language and it consists of 11 header files and 46 source code files. It is designed to be easily portable while still being efficient. Object-oriented programming paradigm is followed and function pointers are utilised heavily. The library contains three different DCT implementations: fixed-point, faster and less accurate fixed-point, and floating-point.

It is quite straightforward to port IJG JPEG library to TMS320VC5510. The header file `jconfig.h` is edited to configure the library but changes to the source code are also needed. The biggest changes are needed in memory management and I/O system. Two allocation functions and two deallocation functions are changed to use OSEck's system calls. IJG JPEG library uses files and related C functions for input and output of JPEG images. As the DSP does not have a file system, images are instead read directly from the memory and they are written directly to the memory. Two new fields are added to

the encoding and decoding object structures of the library. One field is a pointer to the start of the image and the other is the length of the image. Input and output functions are changed accordingly.

JPEG adapter uses IJG JPEG library as is illustrated in the left side of Figure 5.2. First, encoding and decoding objects are created and initialised. After that, the image to be adapted is decoded one scanline, row of pixels, at a time. Conveniently, the JPEG library can scale, subsample, the decompressed image by a given scaling ratio. A ratio of 1/2 is used, as the resolution of the image is wanted to halve. Encoding is done as well one scanline at the time. After the image is decoded and encoded, the objects are destroyed.

```
Allocate and initialize a JPEG decompression object dinfo
Allocate and initialize a JPEG compression object cinfo

Specify the source of the compressed data
Specify the destination for the compressed data

while (scanlines remain to be read) {
    jpeg_read_scanlines(dinfo);
    jpeg_write_scanlines(cinfo);
}

Release the JPEG decompression object dinfo
Release the JPEG compression object cinfo
```

```
int global = 3;
void (*func_pntr) (int *a);

void func_a(int *param)
{
  *param = 5;
}

void func_b()
{
  int local = 0;
  func_pntr = func_a;
  (*func_pntr)(&local);
  global += local;
}
```

Figure 5.2 Usage of IJG JPEG library in JPEG adapter and example of found bug.

The right side of Figure 5.2 illustrates the bug that was found from Texas Instruments' C compiler version 2.57. The functions that read and write scanlines use function pointers to call functions that update the counters that keep track of the current scanline. If the code is compiled with optimisations flags, the compiler does not realise that the function called via a function pointer may update the values of its arguments. In the example code, the value of `global` is 3 instead of 8 after `func_b` is called. Texas Instruments have been informed about the bug and they have promised to fix it (SDSsq29676) in the future releases of the compiler. One reason why a bug this elementary have not been found earlier although the compiler have been used to compile code for millions of processors, is that function pointers or other high-level programming techniques are not traditionally used on DSPs.

6 Testing and measurements

This chapter explains how JPEG adapter was tested and measured. A test client that was implemented to be able to test JPEG adapter is introduced in the first section. The second section describes a real network element to give an idea of what kind of hardware is used to adapt speech in mobile networks. The third section describes the test environment. The hardware and software used for testing are introduced in the detail that is needed to understand the test environment. The fourth section describes what was measured and how. The selection of measurements and the measurement results are also discussed.

6.1 JPEG adapter tester

JPEG adapter tester is a program that runs on a PC and sends adaptation requests to JPEG adapter running on a DSP. The signalling protocol that is used between the two programs was introduced in the previous chapter. Section 6.3 describes an environment where the two programs can be used. JPEG adapter tester was implemented to be able to debug, test, and demonstrate JPEG adapter. During the implementation of JPEG adapter the tester was an invaluable help in ad hoc testing. JPEG adapter tester supports also regression testing and many of the features are useful in measurements. As the name of JPEG adapter tester implies, testing is the fundamental reason for the existence of the program but it is also a convenient tool to demonstrate the features of JPEG adapter.

Despite the diversified roles of JPEG adapter tester, a goal was to keep it simple and easy to use. JPEG adapter tester can be used as a command line tool to execute test cases described in a file. It can be also used as a graphical tool to view the images to be adapted, set the adaptation targets, and view the adapted images. The graphical user interface of the tester is shown in Figure 6.1. It is possible to operate the graphical user interface using only a mouse but keyboard shortcuts are provided for advanced users.

JPEG adapter tester is implemented in Java programming language. This language was chosen because Java programs can be run virtually on any operating system, graphical user interfaces are easy to implement using Java, and because Java contains a wide support for different network protocols. JPEG adapter tester is implemented following the object-oriented programming paradigm.

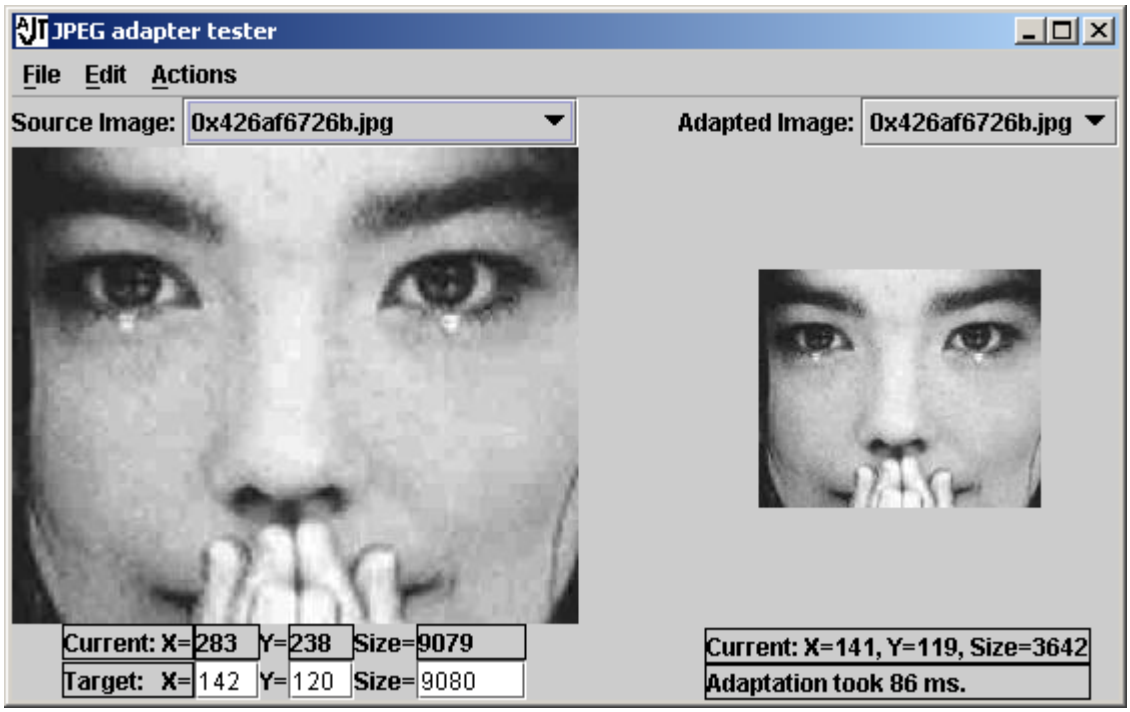


Figure 6.1 Graphical user interface of JPEG adapter tester.

The basic usage of JPEG adapter tester is as follows. When the program is started without command line arguments, a graphical user interface is launched and a thread is created to receive control messages. The image to be adapted, or the source image, is shown in the left side of the user interface. Below that are the input fields for the target values of the adaptation. Choosing Adapt from the Actions menu sends an adaptation request to a DSP. When the DSP has adapted the image, it sends an adaptation complete control message to the tester, which then creates a thread to receive the adapted image and acknowledges the control message. After the adapted image is received, it is shown in the right side of the user interface.

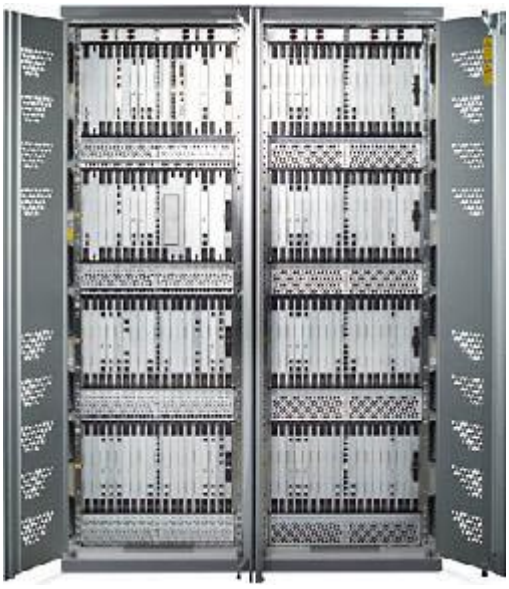
6.2 Media Gateway

Release 4 of 3GPP's UMTS specification was briefly described in Section 2.3. Due to the separation of the control and user planes in the circuit switched core network, a new logical network element called Media Gateway is introduced. Its main tasks are processing user plane traffic and connecting networks based on different transport techniques like IP, time-division multiplexing (TDM), or Asynchronous Transfer Mode (ATM). This section describes Nokia's product called Multimedia Gateway, which implements many logical network elements including Media Gateway. Media elements other than speech are not adapted in this network element.

Multimedia Gateway is basically a big computer with interfaces to different networks and a huge amount of computing power for speech adaptation, or speech transcoding. It is highly scalable and has a hierarchical mechanical structure based on cabinets, subracks, and plug-in units. Multimedia Gateway consists of one to three cabinets. Each cabinet has room for four subracks and each subrack may contain up to 19 plug-in units, which are the basic functional units of the network element. Multimedia Gateway with two cabinets is shown in Figure 6.2.

Plug-in units exist for tasks like speech transcoding, ATM multiplexing, and network interworking. The transcoding units are connected to ATM multiplexers, which are connected to an ATM switch that is the main switching unit of the Multimedia Gateway. The transcoding plug-in unit is described in more detail in Section 6.3.2.

It is easy to understand that power consumption is a crucial property of battery operated mobile devices but it is also important for network elements like Multimedia Gateway. Network elements are usually connected to the national power grid and the problem is not the power consumption as such, as the price of electricity is not an issue in most countries. The problem is that consumed energy turns into heat in the network element. Higher power consumption means larger cooling systems and larger network elements. Floor space is one of the selling points of network elements and companies selling them compete to minimise it.



One cabinet
- Width 600 mm
- Depth 600 mm
- Height 1800 mm

Fully equipped Multimedia Gateway with three cabinets
- Weight ≈ 700 kg
- Power consumption ≈ 10 kW
- Thousands real-time adaptations between PCM and AMR

Figure 6.2 Multimedia Gateway with two cabinets.

6.3 Test environment

6.3.1 Overview

Existing hardware is used to test JPEG adapter. A real network element like Multimedia Gateway is not used, as most of its features would be irrelevant to a program like JPEG adapter. A stripped-down network element containing mainly a transcoding plug-in unit (TPU) is used instead. Because the TPU is not designed to run an IP stack on every DSP it contains, a little tweaking is needed. It is wanted that JPEG adapter and JPEG adapter tester can be run as any programs in the same IP network.

The test environment is illustrated in Figure 6.3. JPEG adapter tester can be run on any PC capable of running Java Virtual Machine. The same PC, or any other PC in the same Ethernet segment, is used to tunnel all the Ethernet frames sent to the Ethernet address of a DSP on the TPU to the host processor of the DSP. The Ethernet frames are tunnelled over UDP/IP and as a result, JPEG adapter running on a DSP sees itself as if it was in the same Ethernet segment as JPEG adapter tester.

JPEG images are transported over UDP as was said earlier. No protocol is used on top of UDP. The sender just divides the image to UDP packets smaller than the maximum transmission unit of the Ethernet and the receiver concatenates the UDP packets. The receiver always knows how many octets to receive and when to stop. UDP is a connectionless protocol and there is no guarantee that the UDP packets will be received in the same order they were sent. This is not a problem in practise because there is no router with packet queues or any other device that could rearrange the packets between JPEG adapter and JPEG adapter tester.

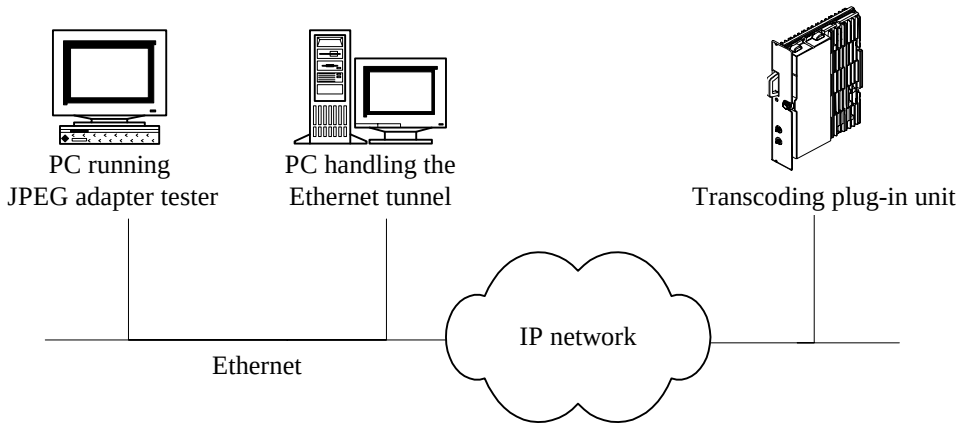


Figure 6.3 Test environment architecture.

6.3.2 Transcoding plug-in unit

Transcoding plug-in unit (TPU) is designed to perform signal processing tasks as efficiently as possible. It contains a large number of DSPs and a smaller number of host processors, whose tasks include handling of the data transport to and from the TPU. The two processors communicate with each other by sharing a region of DSPs memory using the Enhanced Host Port Interface [TI datam]. Figure 6.4 shows the parts of a TPU that are relevant to the testing of JPEG adapter.

Motorola's MPC8260 PowerQUICC II running at 300 MHz is used as a host processor on the TPU that is used to test JPEG adapter. PowerQUICC is Motorola's processor family integrating a PowerPC core and a communications processor module. MPC8260 contains PowerPC G2 core MPC603e. Its communications processor module supports network techniques like Ethernet, TDM, and ATM. MPC8260 consumes power about 3 W. Sun Microsystems' real-time operating system ChorusOS is used on MPC8260.

Each MPC8260 acts as a host for several TMS320VC5510 DSPs. The TPU contains 8 megabytes of external memory for each DSP. This Synchronous Dynamic RAM is accessed using the External Memory Interface. The external memory is considerably cheaper and slower than the internal memory. Both Enhanced Host Port Interface and External Memory Interface can utilise the Direct Memory Access controller to allow movement of data among the different memories without intervention from the CPU.

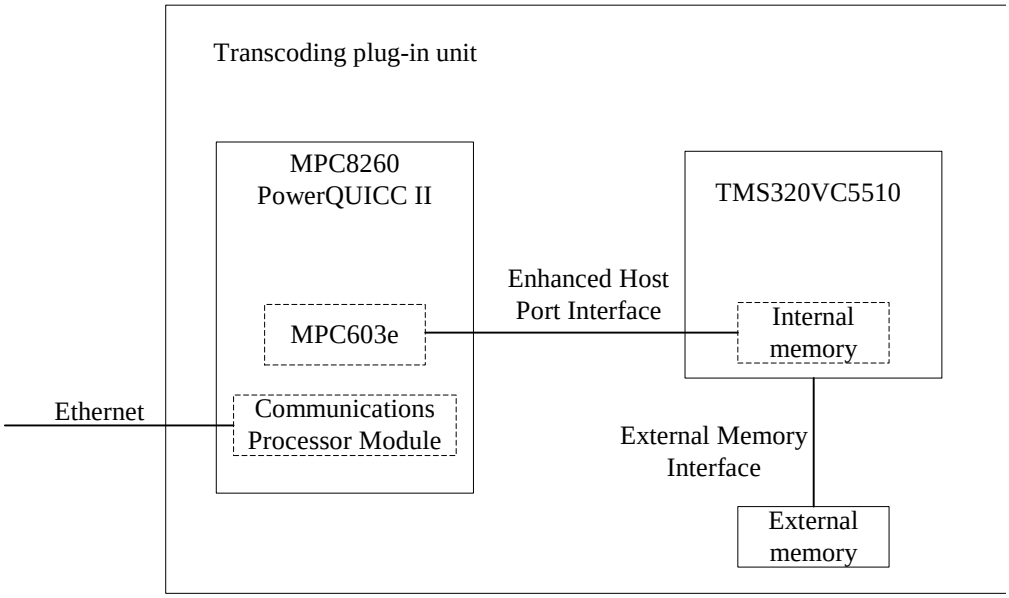


Figure 6.4 A portion of a TPU showing one host and one slave processor.

6.4 Measurements

6.4.1 Overview

JPEG adapter is tested and measured to gain hands-on experience on adapting JPEG images. The aim is to understand how different parameters affect the adaptation time and to find out the portions of the source code where most of the clock cycles are spent. The aim is not to optimise the program but algorithms needing optimisation are tried to identify and the possible optimisation techniques are discussed. Also, measurements on image coding in general, the implementation itself, and the test environment are conducted.

Texas Instruments' integrated development environment called Code Composer Studio is used to debug and measure JPEG adapter. It supports symbolic debugging using either a simulator or an emulator. The simulator can be used to run and debug a program if no real DSP is available. The emulator can be used to debug the TMS320VC5510 via a JTAG interface in real-time without intruding the processor. The profiler functionality of Code Composer Studio is used to profile JPEG adapter. OSEck's system call `get_ticks` is also used to measure JPEG adapter. The system call returns the number of system ticks since system start. The resolution of the system tick is one millisecond.

Five different versions of the same image that was used as an example in Chapter 4 are used in the measurements. Table 6.1 contains some basic information about the five images. Compression ratio is often used to describe compressed images. Another widely used concept is bits per pixel (BPP). The table contains BPP figures for the image files and for the image data. JPEG images consist usually of two parts: headers and image data. The headers contain quantisation, Huffman tables, and some other information.

Table 6.1 Basic information about the images used in the measurements.

Image file	File size octets	Resolution	Pixels	File BPP	Image data octets	Image data BPP
lena32.jpg	1058	32x32	1024	8.3	435	3.4
lena64.jpg	2059	64x64	4096	4.0	1436	2.8
lena128.jpg	4981	128x128	16384	2.4	4358	2.1
lena256.jpg	13330	256x256	65536	1.6	12707	1.6
lena512.jpg	37888	512x512	262144	1.2	37265	1.1

6.4.2 Results and analysis

6.4.2.1 Source code profiling

The source code of JPEG adapter is profiled to find out the algorithms that could be optimised. The results of the profiling are shown in Table 6.2. The tasks in the first column refer to the code sections shown in Figure 5.2. The second column shows how much of the total adaptation time is spent in these three sections. Similarly, the third and fourth columns contain the same information about the two functions inside the while loop. The last two columns contain the final break down of the tasks that were profiled. It can be seen that DCT and entropy coding are the two tasks consuming most of the time as was assumed. It can be also seen that the colour conversions take a lot of time although they are not needed. The resolution can be halved in any colour space and therefore eliminating the conversions will considerably optimise JPEG adapter.

Table 6.2 Source code profiling of JPEG adapter adapting lena256.jpg.

Task	%	Task	%	Task	%
Initialisation	1.4				
While loop	98.5				
		Decoding	56.4		
				Entropy decoding	17
				Inverse DCT	22
				$YC_BC_R \rightarrow$ RGB	10
				Memory zeroing	4
				Other	3.4
		Encoding	42.1		
				RGB $\rightarrow YC_BC_R$	8
				Subsampling	8
				Forward DCT	14
				Entropy encoding	12
				Other	0.1
Finalisation	0.1				

6.4.2.2 Memory usage

JPEG adapter allocates statically one external memory page, 131072 octets, for the source image and another external memory page for the adapted image. An internal memory access takes one clock cycle while an external memory access takes eight clock cycles. It takes 50 clock cycles to copy a byte within the external memory. During the adaptation that was profiled earlier, JPEG adapter itself allocated dynamically 1516 octets of internal memory while the JPEG library allocated 64096 octets of internal

memory and 16472 octets of external memory. Memory was allocated only during the initialisation phase. The stack size of the JPEG adapter process is 2048 octets.

6.4.2.3 Image transport

Image transport over UDP was described in Section 6.3.1. Two parameters affect to the throughput: the frequency used to send the UDP packets and the size of the UDP packets. Typical values that were used during the testing were 40 packets in a second and 960 octets per packet making the data rate 300 kbit/s. No UDP packet was noticed to drop using these values but decreasing the delay between the packets or increasing the size of the packets led to a packet loss. The data rate was tried to increase by making the receiver to acknowledge the received packets and making the sender to send a new packet every time it received an acknowledgement. Due to the long end-to-end delay of the used tunnelling solution, the data rate did not increase significantly. As the achieved data rate of 300 kbit/s is enough for testing, the issue was not investigated further.

6.4.2.4 Image size

The adaptation times of the test images are given in Table 6.3. The adaptation times include only the tasks listed in Table 6.2 and not tasks like image transport. The second column of the table shows the sizes of the images measured in pixels compared to the middle sized image. The third column shows the adaptation times when the fixed-point DCT implementation and the default Huffman tables were used. It can be seen that the adaptation times increase at a slower rate than the image sizes.

Table 6.1 shows that the BPP figures decrease as the image sizes increase. This is not surprising as there is clearly more spatial redundancy in the larger images. DCT computing time does not usually depend on input data while entropy coding does. This partly explains why the adaptation time and image size do not increase at the same rate.

Table 6.3 Adaptation times of different DCT implementations.

Image file	Pixels relative	Fixed-point DCT relative	Floating-point DCT difference %
lena32.jpg	0.06	0.12	+130
lena64.jpg	0.25	0.31	+180
lena128.jpg	1	1	+210
lena256.jpg	4	3.6	+230
lena512.jpg	16	13	+240

6.4.2.5 DCT implementations

TMS320VC5510 does not support floating-point data representation but it is possible to execute floating-point algorithms written in C programming language as the compiler supports floating-point data representation. It compiles every floating-point operation to a series of fixes-point operations that can be executed on the DSP. The last column of Table 6.3 shows the adaptation times using the floating-point DCT implementation compared to the adaptation times using the fixed-point DCT implementation. Although DCT computing takes only 36 % of the adaptation time of the profiled test image, the floating-point DCT implementation increases the adaptation time 230 %. This proves that there is no use in executing floating-point algorithms on a fixed-point processor.

The last column of Table 6.3 affirms the expectation that the relative importance of DCT increases as the spatial redundancy increases.

6.4.2.6 Huffman tables

JPEG images are usually encoded using the Huffman tables that are provided in the annexes of [JPEG]. Also IJG JPEG library uses them by default but it can be used to construct optimal Huffman tables for each image. Table 6.4 contains the results of the measurements using the optimised Huffman tables. The second column shows the adaptation times compared to the adaptation times using the default Huffman tables. The third column shows the differences in the file sizes and the fourth column shows the sizes of the headers. The headers of the test images are 623 octets as can be seen from Table 6.1. The last column shows the differences in the sizes of the image data.

In addition to the longer adaptation times, the use of the optimised Huffman tables increases also the memory usage. The adaptation of the profiled test image using the optimised Huffman tables allocated 47904 octets, or 59 %, more memory.

Table 6.4 Use of optimised Huffman tables compared to the default tables.

Image file	Adaptation time difference %	File size difference %	Headers octets	Image data difference %
lena32.jpg	+67	-44	306	-6
lena64.jpg	+38	-31	326	-3
lena128.jpg	+19	-17	333	-2
lena256.jpg	+12	-8	353	-2
lena512.jpg	+9	-4	356	-2

6.4.3 Future measurements

JPEG adapter is fully implemented in C programming language and one obvious optimisation possibility is to replace some algorithms like DCT with algorithms written in assembly language. Measuring the performance of JPEG adapter using the assembly-optimised algorithms would give a better picture of the capabilities of the processor.

Texas Instruments' Image/Video Processing Library for TMS320C55x was mentioned in Section 5.3 [TI lib]. The function library can be downloaded for free from Texas Instruments' web site. In addition to the functions for the hardware extensions of the processor, the library contains also other assembly-optimised C-callable functions for image and video processing. Useful functions for JPEG adapter include quantisation, dequantisation, colour space conversion, entropy encoding, and entropy decoding.

Use of the library was considered and a plan was to replace some of the functions of IJG JPEG library with the optimised functions. A closer look at the latest version of the library, version 2.30, revealed that it was poorly documented. The source code and the documentation were not consistent and two of the first three functions that were studied more closely contained a bug. For example, the function that computes a reciprocal table uses the register `A1` as a pointer to a temporary storage but never initialises the value of the register. This, of course, leads to an overwriting of data at random address, or at null address as is often the case. The reciprocal table is used in the quantisation of the DCT coefficients. It was decided not to use the library until a fixed and significantly improved version is released.

Another interesting measurement would be the comparison of TMS320VC5510 and some general-purpose processors like Intel's Pentiums. It seems that DSPs are more suitable for media adaptation in the mobile networks than today's GPPs but this should be confirmed by measurements. It is not easy to compare DSPs and GPPs as things like media adaptation performance, power consumption, and price among many other things must be taken into account. It is also important to notice that using processors from the same manufacturer, or even the same processors, in the mobile telephones and networks provides a competitive advantage due to the economics of scale. A company like Nokia sells 150 million mobile telephones in a year but no company use anywhere near that many processors in network elements.

7 Conclusions

The most important service in all telephone networks is the speech service. Different mobile and landline telephone networks use different methods to compress speech. To enable calls between different networks, speech needs to be adapted at the border of the networks. This is done using digital signal processors. As the mobile networks and devices evolve, it is possible to introduce new services. Services based on media elements like images and video require that the transmission speeds of the mobile networks will continue to increase. To enable communication, all media elements need to be adapted to meet the requirements of the communicating parties.

Multimedia messaging service is built on the success of text messaging and Internet e-mail. The service has been in commercial use for over a year but it remains to be seen whether it will ever become as widely used as text messaging. Multimedia messages are composed using a wide set of media types and formats. This thesis concentrated on image coding and especially on the JPEG image format. It was noted that the image compression techniques used in JPEG are partly based on signal processing. Digital signal processors are designed to compute algorithms like discrete cosine transform that is used in JPEG and in many other media formats.

To study the adaptation of JPEG images using a digital signal processor, an application was implemented to halve the resolution of the images. Existing software components were reused and the development of the application was about as straightforward as it would have been to develop the same application for a PC. Profiling the resolution halving of a JPEG image revealed that most of the time was spent on computing discrete cosine transform. Therefore, it seems reasonable to use digital signal processors to optimise the adaptation of JPEG images.

It is not yet clear when the need for media adaptation will be in the same scale as it is today for speech adaptation. Other services in addition to the multimedia messaging service are definitely needed but the introduction of them is only a matter of time as they are just waiting for their turn. Today’s mobile devices and networks are advanced enough for many new services. There does not seem to be any reason why the same hardware based on digital signal processors could not be used to adapt speech and other media elements in future mobile networks.

References

[3GPP 22.034] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects. *High Speed Circuit Switched Data (HSCSD)*. Stage 1. 3GPP TS 22.034 version 5.0.0. June 2002.

[3GPP 22.060] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects. *General Packet Radio Service (GPRS)*. Stage 1. 3GPP TS 22.060 version 5.1.0. March 2002.

[3GPP 22.105] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects. *Services and Service Capabilities*. 3GPP TS 22.105 version 5.1.0. March 2002.

[3GPP 22.140] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects. *Multimedia Messaging Service*. Stage 1. 3GPP TS 22.140 version 5.1.0. March 2002.

[3GPP 23.002] 3rd Generation Partnership Project; Technical Specification Group Services and Systems Aspects. *Network architecture*. 3G TS 23.002 version 3.1.0. September 1999.

[3GPP 23.034] 3rd Generation Partnership Project; Technical Specification Group Core Network. *High Speed Circuit Switched Data (HSCSD)*. Stage 2. 3GPP TS 23.034 version 5.0.0. June 2002.

[3GPP 23.040] 3rd Generation Partnership Project; Technical Specification Group Terminals. *Technical realization of the Short Message Service (SMS)*. 3GPP TS 23.040 version 5.3.0. March 2002.

[3GPP 23.060] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects. *General Packet Radio Service (GPRS)*. Stage 2. 3GPP TS 23.060 version 5.1.0. March 2002.

[3GPP 23.140] 3rd Generation Partnership Project; Technical Specification Group Terminals. *Multimedia Messaging Service (MMS)*. Stage 2. 3GPP TS 23.140 version 5.2.0. March 2002.

[3GPP 26.140] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects. *Multimedia Messaging Service (MMS); Media formats and codecs*. 3GPP TS 26.140 version 5.0.0. March 2002.

[3GPP 43.064] 3rd Generation Partnership Project; Technical Specification Group GERAN. *Overall description of the GPRS radio interface*. Stage 2. 3GPP TS 43.064 version 5.0.0. April 2002.

[Chr00] Christopoulos, C. A.; Ebrahimi, T.; Skodras, A. N. 2000. *JPEG2000: The New Still Picture Compression Standard*.

Proceedings of the 2000 ACM workshops on Multimedia. Pp 45-49. ISBN 1-58113-311-1.

[Eyr00] Eyre, Jennifer; Bier, Jeff. 2000. *The evolution of DSP processors*. IEEE Signal Processing Magazine. Volume 17, Issue 2, pp. 43-51. ISSN 1053-5888.

[Gon01] Gonzalez, Rafael C.; Woods, Richard E. 2001. Digital Image Processing. Prentice Hall. Second edition, 793 p. ISBN 0-13-094650-8.

[Hil01] Hillebrand, Friedhelm. 2001. *GSM and UMTS: The Creation of Global Mobile Communication.* John Wiley & Sons. 590 p. ISBN 0-470-84322-5.

[Hut01] Hutchinson, Jamie. 2001. *Culture, Communication, and an Information Age Madonna*. IEEE Professional Communication Society Newsletter. Volume 45, Number 3. ISSN 0143-433X.

[ISOC] ISO/IEC 9899:1999. *Programming languages — C*. Second edition.

[ITU] International Telecommunication Union. 2002. *World Telecommunication Development Report.* ISBN 92-61-09831-2.

[JFIF] Hamilton, Eric. *JPEG File Interchange Format*. September 1992.

[JPEG] ISO/IEC 10918-1. *Digital compression and coding of continuous-tone still images: Requirements and guidelines*. February 1994.

[Kaa01] Kaaranen, Heikki et al. 2001. *UMTS Networks: Architecture, Mobility and Services*. John Wiley & Sons. 326 p. ISBN 0-471-48654-X.

[Lam01] Lam, Calvin K.M.; Tan, Bernard C.Y. 2001 *The Internet is changing the music industry*. Communications of the ACM. Volume 44, Issue 8, pp 62-68. ISSN 0001-0782.

[libjpeg] Lane, Thomas G. 1998. *Documentation and source code of the Independent JPEG Group's JPEG software*. Version 6b.

[Loe89] Loeffler, Christoph; Ligtenberg, Adriaan; Moschytz, George S. 1989. *Practical fast 1-D DCT algorithms with 11 multiplications*. International Conference on Acoustics, Speech, and Signal Processing. Volume 2, pp. 988-991.

[Mof01] Moffitt, Jack. 2001. *Ogg Vorbis—Open, Free Audio—Set Your Media Free*. Linux Journal. Volume 2001, Issue 81. ISSN 1075-3583.

[OMA conf] Open Mobile Alliance. *MMS Conformance Document*. Version 2.0.0. February 2002.

[OMA arch] Open Mobile Alliance. *Multimedia Messaging Service Architecture Overview*. Version 1.1. November 2002.

[OMA ctr] Open Mobile Alliance. *Multimedia Messaging Service Client Transactions*. Version 1.1. October 2002.

[OMA enca] Open Mobile Alliance. *Multimedia Messaging Service Encapsulation Protocol*. Version 1.1. October 2002.

[OMA UAPr] Open Mobile Alliance. *User Agent Profile*. Version 2.0. May 2003.

[OSE] Enea OSE Systems. *OSE for DSP: User's Guide*. 2001.

[Pee00] Peersman, G. et al. 2000. *A tutorial overview of the short message service within GSM*. Computing & Control Engineering Journal. Volume 11, Issue 2, pp 79-89. ISSN 0956-3385.

[RFC 2045] Internet Engineering Task Force. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. IETF RFC 2045. November 1996.

[RFC 2046] Internet Engineering Task Force. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. IETF RFC 2046. November 1996.

[RFC 2387] Internet Engineering Task Force. *The MIME Multipart/Related Content-type*. IETF RFC 2387. August 1998.

[RFC 2616] Internet Engineering Task Force. *Hypertext Transfer Protocol -- HTTP/1.1*. IETF RFC 2616. June 1999.

[RFC 2822] Internet Engineering Task Force. *Internet Message Format*. IETF RFC 2822. April 2001.

[Roc00] Rockmore, Daniel N. 2000. *The FFT: an algorithm the whole family can use*. Computing in Science & Engineering. Volume 2, Issue 1, pp 60-64. ISSN 1521-9615.

[Rod03] Rodermund, Friedhelm. 2003. *A Picture Speaks a Thousand Words – From SMS to MMS*. World Markets Research Centre Business Briefing: Wireless Technology 2003. Pp 86-89.

[Sav01] Savolainen, Petri. 2001. *Interfacing digital signal processors with a limited IP Stack*. Master's Thesis. Helsinki University of Technology. 111 p.

[Shi99] Shi, Yun Q.; Sun, Huifang. 1999. *Image and Video Compression for Multimedia Engineering*. CRC Press. 480 p. ISBN 0-8493-3491-8.

[Teo03] Teo, Hock Hai; Wang, Hao; Xu, Heng. 2003. *Foundations of SMS commerce success: lessons from SMS messaging and co-opetition*. Proceedings of the 36th Annual Hawaii International Conference on System Sciences. Pp 90-99.

[TI datam] Texas Instruments. *TMS320VC5510 Fixed-Point Digital Signal Processor Data Manual*. February 2003.

[TI hwexs] Texas Instruments. *TMS320C55x Hardware Extensions for Image/Video Applications Programmer's Reference*. February 2002.

[TI lib] Texas Instruments. *TMS320C55x Image/Video Processing Library Programmer's Reference*. March 2003.

[W3C SMIL] World Wide Web Consortium. *Synchronized Multimedia Integration Language*. Version 2.0. August 2001.

[Wal92] Wallace, G.K. 1992. *The JPEG still picture compression standard*. IEEE Transactions on Consumer Electronics. Volume 38, Issue 1, pp. 18-34. ISSN 0098-3063.

[WAP WSP] Wireless Application Protocol Forum. *Wireless Session Protocol Specification*. Version WAP 1.2.1. May 2000.