Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto
Tietoliikennetekniikan laitos

Diplomityö

Unlicensed Mobile Access Concept from Testing Point of View

Toni Antila
0074750 TiteN
antila@lut.fi
0405595841

Helsinki 07.01.2006

Toni Antila

Tiivistelmä

Lappeenrannan teknillinen yliopisto

Tietotekniikan osasto

Toni Antila

Unlicensed Mobile Access - käsite testauksen näkökulmasta

Diplomityö

2006

69 sivua, 27 kuvaa, 5 taulukkoa, 3 liitettä

Tarkastajat: Prof. Arto Kaarna, DI Jukka Talvitie

Ohjaaja: Ins. Jani Raussi

Hakusanat: GSM, UMA, CVOPS, protokolla, testaus, mobiili

Viime vuosikymmenien aikana kommunikaatioteknologiat ovat kehittyneet erittäin paljon. Uusia verkkoja, liityntätekniikoita, protokollia ja päätelaitteita on luotu alati kehittyvällä vauhdilla, eikä hidastumisen merkkejä ole näkyvissä. Varsinkin mobiilisovellukset ovat kasvattaneet markkinaosuuksiaan viime aikoina. Unlicensed Mobile Access (UMA) on uusi liityntätekniikka mobiilipäätelaitteille, joka mahdollistaa liitynnän GSM-runkoverkkoon WLAN- tai Bluetooth – tekniikoiden avulla. Tämä diplomityö keskittyy UMAan liittyviin teknologioihin, joita tarkastellaan lähemmin ensimmäisissä kappaleissa. Tavoitteena on esitellä, mitä UMA merkitsee, ja kuinka eri tekniikoita voidaan soveltaa sen toteutuksissa.

Ennen kuin uusia teknologioita voidaan soveltaa kaupallisesti, täytyy niiden olla kokonaisvaltaisesti testattuja. Erilaisia testausmenetelmiä sovelletaan laitteiston ja ohjelmiston testaukseen, mutta tavoite on kuitenkin sama, eli vähentää testattavan tuotteen epäluotettavuutta ja lisätä sen laatua. Vaikka UMA käsittääkin pääasiassa jo olemassa olevia tekniikoita, tuo se silti mukanaan uuden verkkoelementin ja kaksi uutta kommunikaatioprotokollaa. Ennen kuin mitään UMAa tukevia ratkaisuja voidaan tuoda markkinoille, monia erilaisia testausmenetelmiä on suoritettava, jotta varmistutaan uuden tuotteen oikeasta toiminnallisuudesta.

Koska tämä diplomityö käsittelee uutta tekniikkaa, on myös testausmenetelmien yleisen testausteorian käsittelemiselle varattu oma kappale. Kappale esittelee erilaisia testauksen näkökulmia ja niihin perustuen rakennetaan myös testausohjelmisto. Tavoitteena on luoda ohjelmisto, jota voidaan käyttää UMA-RR protokollan toiminnan varmentamiseen kohdeympäristössä.

Abstract

Lappeenranta University of Technology

Department of Information Technology

Toni Antila

Unlicensed Mobile Access Concept from Testing Point of View

Master's thesis

2006

69 pages, 27 figures, 5 tables, 3 appendices

Examiners: Prof. Arto Kaarna, DI Jukka Talvitie

Instructor: Ins. Jani Raussi

Keywords: GSM, UMA, CVOPS, protocol, testing, mobile

Over the last few decades the communication technologies have advanced a great deal. New networks, access methods, protocols and user-equipments have been created with ever-increasing speed and the pace does show any signs of slowing down. Especially mobile solutions have increased their market share over the past few years. Unlicensed mobile access (UMA) is a new access method for mobile stations that enables the usage of core GSM network through WLAN or Bluetooth access technologies. This thesis work concentrates on the technologies behind the UMA concept and they are being studied at first chapters. The goal is to show what UMA generally means and how different technologies are applied with UMA.

Before new technologies can be applied to any commercial purposes, they must be comprehensively tested. Different testing methods apply for software and hardware components but the overall goal of testing is the same, which is to reduce the uncertainty and increase the quality of a product. Although unlicensed mobile access concept uses mainly existing technologies and protocols, it still brings in two new protocols. Before any UMA supporting solution that can be brought to markets, many different kinds of testing methods must have been carried out to ensure the correct functionality of the new product.

As this thesis work deals with new technology, the testing methods and general testing theory is revieved in an own chapter. The chapter presents different testing aspects and based on those aspects the software testing implementation is constructed. The purpose is to create a protocol testing software that can be used in protocol conformance verification of UMA-RR protocol in specified environment.

Table of contents

1. INTRODUCTION 1
1.1 Scope of the thesis 1
1.2 Structure of the thesis 2
2. UNLICENSED MOBILE ACCESS CONCEPT 4
2.1 Architecture of UMA 7
2.2 Interfaces 8
2.3 Access point and access technologies 10
2.3.1 The 802.11 standard family 11
2.3.1.1 802.11 protocol stack 11
2.3.2 Bluetooth 15
2.3.2.1 Bluetooth protocol stack 17
2.3.2.2 PAN profile 19
2.3.2.3 Comparison between 802.11 and Bluetooth 20
2.3.3 Using Access Points 21
2.4 The UMA Network Controller 22
2.4.1 Discovering of and registering to UMA network 23
2.5 UMA specified protocols 24
2.5.1 UMA Radio Resource protocol 24
2.5.2 UMA Radio Link Control 26
2.6 Security in UMA 28
2.6.1 IPSec 29
2.6.1.1 Authentication Header and Encapsulating Security Payload 30
2.6.1.2 Internet Key Exchange Version 2 33
2.6.2 Extensible Authentication Protocol 35
3. TESTING 37
3.1 Testing methods and aspects 38
3.1.1 Different levels of testing 38
3.1.2 The V-model of testing 39
3.1.3 Test planning 40
3.1.4. Black-box testing 42
3.1.4. White-box testing 42
3.1.4. Hybrid methods 43
3.2 Protocol testing 43
4. PROTOCOL TESTING IMPLEMENTATION 48
4.1 CVOPS 49
4.1.1 Virtual tasks 49
4.1.2 Primitives, PDUs and frames 52
4.1.3 File identifiers 52
4.2 UMA-RR CVOPS testing implementation 54
4.2.1 States and timers 55
4.2.2 Messages and message sequences 57
4.2.3 Encoding and decoding functions 59
4.3 Test case execution 63
4.3.1 Conformance verification 64
4.4 Analysis of results and implementation 65
5. CONCLUSIONS 67

SOURCES **69**
APPENDICES

Abbreviations

A	A – Interface
AAA	Authentication, Authorization and Accounting
ACL	Asynchronous Connection-Less
AH	Authentication Header
AP	Access Point
BSC	Base Station Controller
BSSAP-LE	Base Station System Application Part LCS Extension
CCK	Complementary Code Keying
CID	Cell Identity
CN	Core Network
CS	Circuit Switched
CTS	Clear To Send
CVOPS	C Virtual Operating System
DSSS	Direct Sequence Spread Spectrum
ESP	Encapsulating Security Payload
ETSI	European Telecommunications Standards Institute
FDM	Frequency Division Multiplexing
FHSS	Frequency Hopping Spread Spectrum
FMC	Fixed-Mobile Convergence
GERAN	GSM-Edge Radio Access Network
GSM	Global System for Mobile Communication
HLR	Home Location Register
HPLMN	Home Public Land Mobile Network
HR-DSSS	High Rate Direct Sequence Spread Spectrum
IEEE	Institute of Eletrical & Electronics Engineers
IETF	Internet Engineering Task Force
IKEv2	Internet Key Exchange Version 2
IMS	IP-based Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
LAI	Location Area Identity
Lb	Lb - Interface
LLC	Logical Link Control
MAC	Medium Access Control
MAP	Mobile Application Part

MCC	Mobile Country Code
MNC	Mobile Network Code
MS	Mobile Station
MSC	Mobile Switching Centre
NAP	Network Access Point
NAV	Network Allocation Vector
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open Systems Interconnection
PAN	Personal Area Network
PANU	Personal Area Network User
PS	Packet Switched
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RR	Radio Resource
RTS	Request To Send
SA	Security Associations
SCO	Synchronous Connection Oriented
SDP	Service Discovery Protocol
SGSN	Serving GPRS Support Node
SGW	Secure Gateway
SIP	Session Initiation Protocol
SMLC	Serving Mobile Location Centre
TBF	Temporary Block Flow
TCP	Transmission Control Protocol
TFI	Temporary Flow Identifier
TKIP	Temporal Key Integrity Protocol
UMA	Unlicensed Mobile Access
UMAN	Unlicensed Mobile Access Network
UMTS	Universal Mobile Telecommunication System
UNC	UMA Network Controller
Up	Up - Interface
VLR	Visited Location Register
VoIP	Voice over Internet Protocol
VPLMN	Visited Public Land Mobile Network
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access

1. INTRODUCTION

Fixed-mobile convergence (FMC) has recently been a very discussed trend in the telecommunication business. Convergence simply means a phenomenon in which two or more existing technologies, markets, producers, boundaries, or value chains combine to create a new force that is more powerful and efficient than the sum of its parts. In FMC this means the integration of wireline and wireless technologies and services to create a single telecommunications network foundation.

Unlicensed mobile access (UMA) is one of the new access technologies that have been seen as one step closer to FMC. It is a wireless access method that makes it possible to connect to GSM core network using Bluetooth or WLAN (802.11) access technologies. The goals of UMA are to enhance customer premises coverage, increase network capacity and potentially lower costs [UMAu05].

The Unlicensed Mobile Access standard is defined by the UMA consortium, which currently consists of 14 companies including e.g. Nokia, Motorola, Sony Ericsson and Siemens. The UMA consortium is also working with the 3GPP to use the specification as the basis for the development of a formal standard. At this point there have been developed some solutions for UMA supporting mobile phones and for network elements that UMA requires.

1.1 Scope of the thesis

This thesis work concentrates on UMA technology and also reviews it from the testing point of view. The structure of this thesis work can be divided into two parts. First part introduces the UMA concept and its architecture. Also used protocols, access technologies and security are considered. The second part of this work concentrates on general aspects of software testing, implementation of a protocol testing program, gathering essential information from the results and making conclusions of the subject matter.

Testing is a very important issue when new software technologies are created and especially protocol testing, as the protocols are the foundation that define the basic services for upper layer software and components. In this thesis work the protocol testing is considered along with the general testing theory in the second part. An implementation of a testing program that simulates a protocol used in UMA is made with CVOPS.

This implementation is used to test the basic signalling sequences. The idea is to create a testing implementation that is at some level portable or can be used together with other testing implementations. Secondary value-adds received from this thesis work could be considered the increased competence knowledge of new UMA technology and also knowledge of CVOPS environment and protocol testing programming.

1.2 Structure of the thesis

Second chapter introduces general information of UMA technology. Most of the information presented in second chapter is based on specifications created by the UMA consortium. These specifications are free to download from their webpage, which address can be found at [UMAa05]. Specifications are comprehensive and they elaborate the requirements of UMA technology well. Information of existing protocols, elements and techniques are not presented in specifications as they are defined elsewhere. The specifications concentrate on the new concepts that UMA requires. Main purpose of the second chapter is to introduce the technology so that the later chapters of testing and its implementation concerning UMA can be understood more easily.

Third chapter concentrates on general aspect of testing. Information of different testing methods is detailed and some relevant aspects of testing procedures are presented.

Fourth chapter describes the testing implementation made for this thesis work. Testing has been recognized to be very important as new technologies are introduced. Nowadays the mobile communication systems have grown to be very complex so it is vital to test new features, elements, protocols and services very delicately. In this thesis work the basic idea of the testing implementation is to test the protocol used between mobile station and the UMA network controller. Lower layers of the UMA protocol stack use other well-defined

protocols such as TCP and IP. These protocols have been tested earlier and found to be working so the implementation in this work concentrates on testing procedures of the new protocol specified in UMA specifications.

2. UNLICENSED MOBILE ACCESS CONCEPT

Unlicensed mobile access is a method that provides new access techniques to core network for mobile phones. UMA technology enables access to mobile voice and data services over IP broadband technologies and unlicensed spectrum technologies. Service providers can enable subscribers to roam between cellular networks and IP networks. Access points have a support for WLAN or Bluetooth access techniques. At first stage of UMA, the mobile devices support both access methods of existing GSM access through base transceiver station and unlicensed mobile access through IP-based broadband network. Both access methods end up using the same GSM core network, so the differences are in radio access methods and network elements directly after radio access and broadband IP network.

UMA introduces a new gateway element called UMA Network (UMAN). Implementation and usage of UMAN requires no changes to existing GSM core network. UMAN acts as a gateway between IP-network and GSM core network. Basic operations for this network element are to support protocol conversion between different networks and also to provide authentication and security methods for mobile user. Mobile stations that can use UMA access are dual-mode devices that support also traditional radio access through base transceiver station. The following Figure 1 shows the basic concept of how UMA technology works.

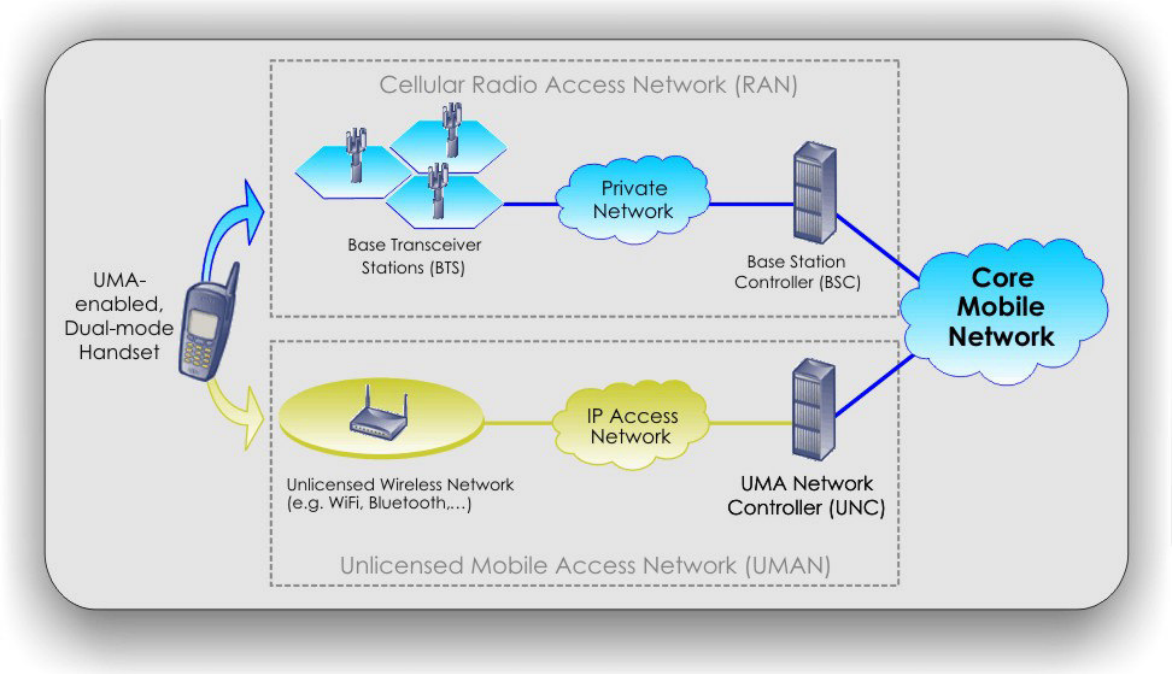

Figure 1. The general concept of UMA, from [UMAo05]

UMAN consist of two operational parts, UMA network controller (UNC) and security gateway (SGW). Both of these elements are illustrated in more detailed level in later chapters. UNC can be seen to core network as GERAN base station subsystem in GSM network. The SGW is used to terminate secure remote access tunnels from the MS and also to provide authentication and encryption for signalling and user plane traffic.

Benefits gained from UMA technology vary between user groups. Some scenarios of different user groups and benefits of UMA are listed in [UMAu05]. As mentioned in scenarios, some users appreciate a better usability, as they would prefer one integrated method for telecommunications instead of having to use different devices for fixed-line and mobile communication. Others may suffer from bad connection when residing inside buildings and want to use a different access method, which could enhance the connection quality at indoors. Nevertheless, the one important benefit can be said to be improved data transfer rates, which compared to the GSM - or UMTS - systems, are significantly better at this point. Next table shows typical data transfer rates for GSM with EDGE, UMTS and UMA access methods.

Table 1. Data transfer rates of different mobile technologies [sig01a],[Sys00]

Technology	Data transfer rate
GSM + EDGE	Theoretical 473,6 kbps, **386 kbps** in use
UMTS	Theoretical 2 Mbps, **386 kbps** currently
Bluetooth	**723 kbps / 2 Mbps** currently
802.11a	**54 Mbps** defined in standard
802.11b	**11 Mbps** defined in standard

As the upper table shows, current data transfer rates are quite much better with UMA when using Bluetooth or 802.11 access methods. However, the standard of UMTS is developing and future transfer rates are predicted to reach 2 Mbps rate, which is comparable to Bluetooth. Also it is noticeable that the actual transfer rates in 802.11 are not always as high as mentioned in the specifications as the bandwidth of one access point is divided between all the users in that base station area. The other important aspect in comparison is the operational range, which is, compared to GSM access to GERAN, significantly smaller with maximum of few hundred meters. [Ora00] [Sys00]

2.1 Architecture of UMA

This chapter presents basic information and important elements of UMA so that the general idea of UMAs architectural design can be clarified. UMAs architecture is described in more detailed level in UMA specifications [UMAa05].

Architecture of UMA consists of a dual-mode mobile station, one or more access points (AP), broadband IP network and one or more UMA network controllers (UNC). The term dual-mode in this case means both traditional GSM-based access via GERAN and also new access method using 802.11 or Bluetooth. Access points are used to establish a wireless connection between mobile subscriber and broadband IP network. On the other side of IP network resides the UNC, which acts as a gateway between the IP network and the GSM core network. The GSM core network doesn't require any changes with UMA as it uses the same protocols with both UNC and GERAN BSC. To clarify some architectural points, the Figure 2 presents the UMAs functional architecture.

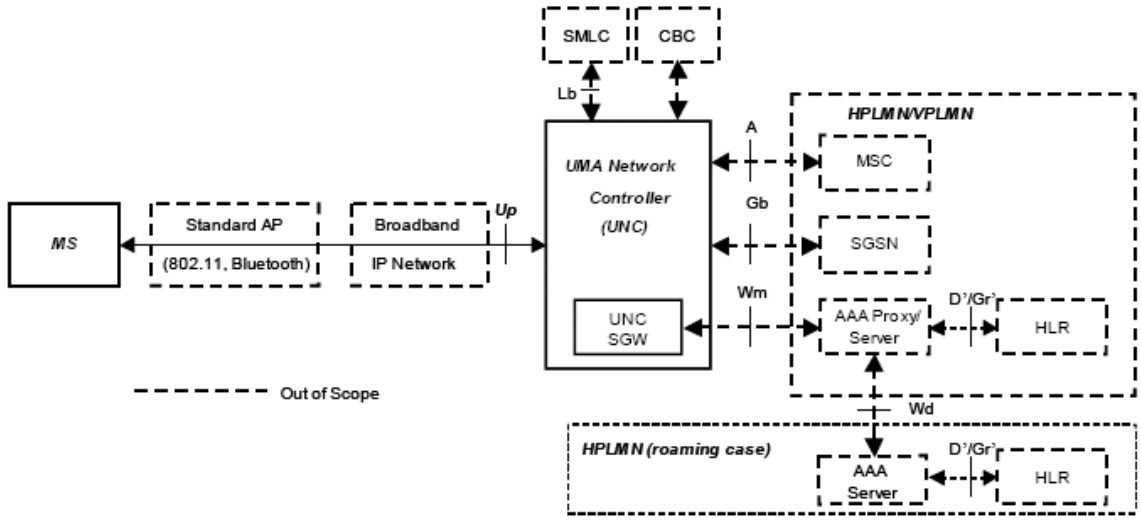

Figure 2. UMAs functional architecture [UMAa2005]

The figure 2 shows all the necessary elements and interfaces of the UMA concept. Parts that are out of scope of the specifications are such networks and elements that don't require any particular changes when applying UMA access method. Mobile stations access a

broadband IP network through standard access point. Currently supported access methods are 802.11 or Bluetooth, which both are standardized and widely supported. Signalling and user plane traffic is carried through IP network to UNC, which handles the connection between IP network and GSM core network. As the figure shows, different interfaces are used with UNC, depending on which side the signalling or user plane data comes from. Next chapters review some basic information of the used interfaces, access points and access technologies.

2.2 Interfaces

A-, Gb-, Wm-, Lb-, and D'/Gr'- interfaces are standardised and commonly used in 2G/3G mobile networks. They define the used protocols and methods for interoperation between different network elements. These interfaces are the basis of interaction between network elements as they define common physical interconnection characteristics, signal characteristics and meanings of exchanged signals. In UMA concept, all of the protocols used in interfaces, are supported by the UNC, as the UNC makes a conversion between 2G/3G network based mobile communication protocols and the IP network based protocols.

The A interface works between the base station controller (BSC) and the mobile switching centre (MSC). Speech and data are transmitted digitally over PR (PCM-30) systems based on the ISDN standard [ITUG.732]. A PCM-30 system has 30 full-duplex channels that operate with the rate of 64 kbit /s. Total transmission rate is 2,048 Mbit / s in full-duplex mode. Two channels are required for synchronizing and signalling. Two primary protocols for this interface are DTAP (Direct Transfer Application Part) and BSSMAP (Base Station System Management Application Part).

The Gb interface is used between the BSC and the SGSN (serving GPRS support node). Main function for this interface is to define the used protocols and techniques needed for support of GPRS packet traffic and signalling. Gb interface uses a high-speed Frame Relay link that is built on E1 or T1 connection. The most important protocol in Gb interface is the

BSSGP (Base Station System GPRS protocol). This protocol provisions the radio related information and operates the node management control functions. [And01]

The Wm and Wd are used with the AAA server and the UNC. Wm is used for relaying authentication and authorization messages between AAA Proxy / Server and UNC-SGW. Wd is used in roaming cases where the mobile subscriber is not in HPLMN (Home Public Land Mobile Network) for relaying authentication and authorization information between visiting AAA and home AAA.

The D' / Gr' interface is used for exchanging subscription information between 3G AAA (Authentication, Authorization and Accounting) and the HLR (Home Location Register) by means of MAP (Mobile Application Part) protocol. This interface is based on two different interfaces depending on if it's used between 3G MSC and HLR or between SGSN and HLR.

The Lb interface is the newest addition to specifications. It is used between the SMLC (Serving Mobile Location Centre) and UNC. Lb interface is used to perform location connection oriented information exchange. Two important protocols used in Lb are BSSAP-LE [TS09.31] and BSSLAP [TS08.71]. However these protocols are rarely used as current UMA specifications make only usage of SMLC with emergency call location determination.

Up interface is the new interface that UMA requires and it is used between MS and UNC. This interface is used to define the signalling, voice and data transfer protocol architectures for both circuit and packet switched connections. Up interface operates over an IP transport network and it relays GSM/GPRS signalling and traffic between the core network and the MS. Up interface uses many common protocols, like TCP or UDP, that are standardized and widely used. Up also uses two new protocols that are defined for the UMA standard. They are the UMA-RR and the UMA-RLC. These new protocols are the most important protocols concerning this thesis work and they are presented in more detail at chapters' 2.5.1 and 2.5.2.

2.3 Access point and access technologies

Access point (AP) provides a radio link between the mobile station and the IP network using an unlicensed mobile spectrum. AP operates on OSI - models physical and data link layers. Physical layer sets the mechanical, electrical and timing interfaces for the physical transmission medium. In this case the transmission medium is wireless and the data is transferred with the usage of electromagnetical signals. Data link layer provides the functional and procedural means to enable the moving of data across the physical network elements. It accomplishes this task by separating the transmitted input data into data frames. Frames are transmitted sequentially into receiving element(s). Data link layer has also mechanisms to detect and sometimes correct errors that occur in the physical layer.

Currently the UMA specifications support two types of access methods, which are the WLAN 802.11x standard family defined by the IEEE and the Bluetooth defined by the Bluetooth Special Interest Group (SIG). Both of these methods are commonly used in many situations where wireless access and data transfer between different network devices is needed. Also there is a continuous development for making improvements and new versions of these standards. The UMA specifications set some requirements for minimum radio performance and also some recommended practises. These radio performance requirements are:

For Bluetooth-based access point:

- Transmit power of the antenna input for AP should be +20 dBm (+0/-3 dBm)
- Receive sensitivity should be at least -86 dBm
- Antenna gain should be at least -0 dBi

For 802.11-based requirements:

- Transmit power at the antenna input should be +17 dBm (+3/-2 dBm)
- Receive sensitivity should be at least -87 dBm
- Antenna gain should be at least -0 dBi

Other access methods are not mentioned in current specifications so at least the first version of UMA support only these two. Next chapters concentrate on the two standards and present the basic information about their architecture and operational methods.

2.3.1 The 802.11 standard family

802.11 refers to family of specifications developed by the IEEE. As the 802.11 technology is in close relation to the Ethernet (802.3), it was initially referred as *The* W*ireless Ethernet.* Later the term Wi-Fi has been generally adopted to mean the 802.11 techniques. The first version of the standard was developed in 1997 and today there are four basic versions of standard and other supplementary standards for e.g. security or national issues in addition. All these versions have a separate working group that keeps up the defining of specifications.

2.3.1.1 802.11 protocol stack

The protocol stack used by all the 802.11 variants has a certain commonality of structure. The physical layer of 802.11 corresponds the physical layer in OSI model well, but the data link layer is divided into two separate sublayers. In 802.11 they are the Medium Access Control (MAC) sublayer and the Logical Link Control (LLC) sublayer. A partial view of the protocol stack of 802.11 is given in the next figure. The Figure 3 shows the separation of data link layer and also some standards of the 802.11 family.

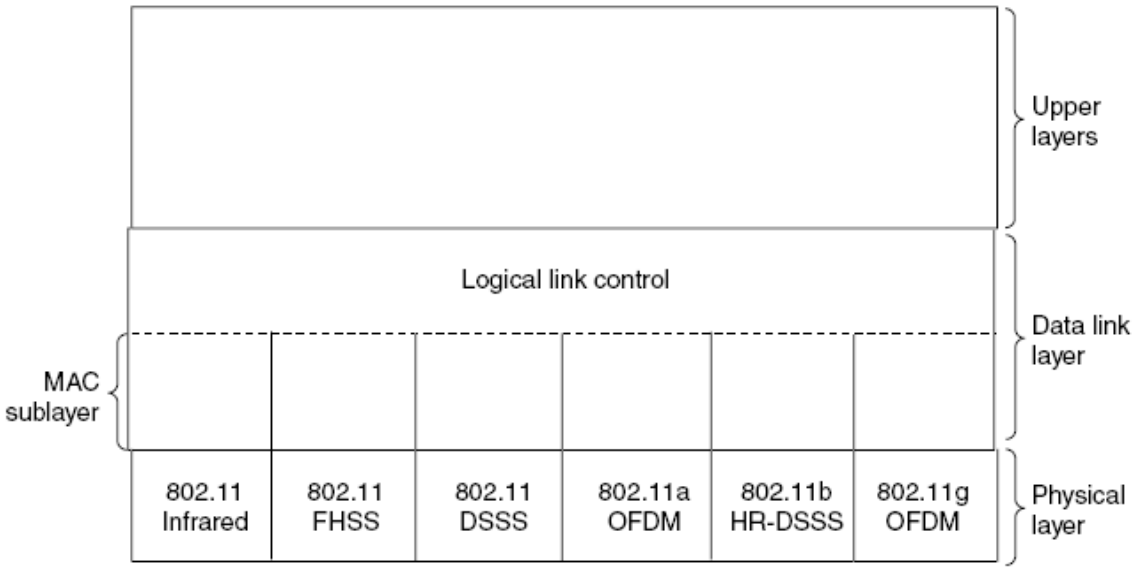

Figure 3. Partial view of the 802.11 protocol stack [Tan03]

The physical layer in the previous partial protocol stack has five different transmission techniques. They can be divided into to main types, namely low power radio frequency (RF) and infrared (IR). Types that use RF methods can then be divided into four classes, depending on the used modulation method. The IR has two subclasses, which are direct beam systems and diffused beam systems. The infrared systems are however rarely used. The radio frequency technologies are multicarrier systems that transmit a high-rate data stream by splitting it to lower-rate streams, which are transmitted simultaneously over multiple orthogonal subcarriers. This technique was developed to fulfil requirements of high data rates.

Frequency Hopping Spread Spectrum (FHSS) uses 79 channels that are 1 MHz wide each. They operate on the frequency between 2,4 - 2,483 GHz. FHSS takes the data signal and modulates it with a carrier signal that hops from frequency to frequency as a function of time. A pseudorandom number generator is used to produce the sequence of frequencies. The amount of time spent at each frequency can be adjusted, but it must be less than 400 ms. Randomization provides a fair way to allocate spectrum and also provides increased security since an intruder who doesn't know the hopping sequence or timing cannot eavesdrop on transmissions. The transfer speed of FHSS is restricted to 1 Mbps or 2 Mbps. [Wal02]

Direct Sequence Spread Spectrum (DSSS) is another modulation method that is used in 802.11. In DSSS a data signal at the sending station is combined with a higher data rate bit sequence, or chipping code, that divides the user data according to a spreading ratio. Each bit is transmitted as 11 chips, using a Barker Sequence. The binary adder effectively multiplies the length of the binary stream by the length of the sequence. This increases the signalling rate and makes the signal span a greater amount of frequency bandwidth. DSSS also has operational transfer rates of 1 Mbps or 2 Mbps. [Wal02]

Orthogonal Frequency Division Multiplexing (OFDM) is a transmission technique based upon the idea of frequency-division multiplexing (FDM) where multiple signals are sent out at different frequencies. OFDM delivers up to 54 Mbps transfer rate and uses 52 different frequencies in the 5 GHz band. In OFDM each signal is assigned a different frequency (subchannel) within the main channel and transmitted out of phase with other signals. This has advantages as immunity to narrowband interference and the possibility of using non-contiguous bands.

High Rate Direct Sequence Spread Spectrum (HR-DSSS) is an enhanced version of DSSS. HR-DSSS offer better transfer rates of 5,5 Mbps and 11 Mbps. To provide the higher rates, 8-chip complementary code keying (CCK) is employed as the modulation scheme. The chipping rate is 11 MHz, which is the same as the DSSS system. HR-DSSS also uses complex set of Walsh/Hadamard functions to create a CCK.

In 802.11 the MAC sublayer is used to determine how the channel is allocated by coordinating access to a shared radio channel and utilizing protocols that enhance communications with two or more 802.11 elements. MAC sublayer uses physical layer to perform transmitting and receiving of frames and also tasks of carrier sensing.

With wireless communication the stations may not always be within radio range of each other. This could cause a so-called hidden station problem. Figure 4 illustrates this problem. Station C is out of a range of station A and therefore cannot hear the transmission of station A. If station A transmits to station B it is possible that also station C is

simultaneously transmitting to station B. This could result a collision of frames in the range of station B.

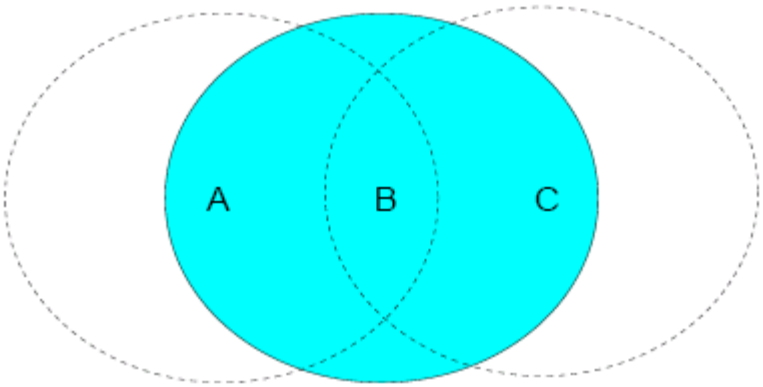

Figure 4. Hidden station problem

To deal with this kind of problem, the 802.11 supports Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) protocol. In this protocol, both physical and virtual channels of sensing are used. Physical channel sensing means that when station wants to transmit, it first listens the channel. If the channel is idle, it starts transmitting and transmits the whole frame at a time. With this there still might be interference at the receivers side and a collision could occur. In this case the transmitting party waits a random time based on a binary exponential backoff algorithm and tries to transmit again. [Wal02]

Virtual channel sensing is based on the sending of special frames that are used for requesting a permission to send data. When a station wants to send data to another, it first sends an RTS (request to send) frame requesting permission for data transfer. When the receiving side receives this request, it may grant a permission, in which case it sends a CTS (clear to send) frame back. Upon receipt of the CTS the initiating station sends a data frame and starts an ACK timer. When the receiving side has received a data frame it sends back an ACK frame, which terminates the exchange. Other station within the range of the sender or the receiver can receive either RTS or CTS frame. If they do, they realize that other stations are going to send data soon and they abstain on sending data at that same time. Other stations will set their virtual carrier sense indicator that is called NAV (Network Allocation Vector) for a given duration. The duration is estimated from the sent RTS or CTS frame.

The strength of the 802.11 standard is its already firm stand among the wireless standards and also in the wireless markets. Also the data transfer rates are good compared to 2G or 3G mobile communications methods. Negative things when applying 802.11 with UMA supporting devices could be the power consumption and also higher costs with 802.11 cards. Also the UMA standard doesn't make a stand of which version of 802.11 standard should be used in mobile stations and access points. This could cause extra costs and need for different kinds of access points, as some versions of the 802.11 standards are not compatible with each other.

2.3.2 Bluetooth

The other access method, that UMA supports, is the Bluetooth. First version of the Bluetooth standard was developed in 1994, when Ericsson wanted to create a method for connecting their mobile phones into other devices, like PDAs. Together with Nokia, IBM, Intel and Toshiba, it formed a Special Interest Group (SIG) that developed a wireless standard for seamless short-range interconnection between both mobile and stationary devices. Bluetooth chips were designed to be low power and inexpensive, so that they could be used in wider range of appliances.

The architecture of Bluetooth is based on piconets and scatternets. Piconet is a system that consists of a master node and up to seven active slave nodes. Multiple piconets can exist in a same space and they can be connected to each other via a bridge node. Interconnection of piconets is called a scatternet. The following Figure 5 shows the idea of piconets and scatternets. Slave 4 is serving as a bridge between two piconets, which are controlled by masters A and B.

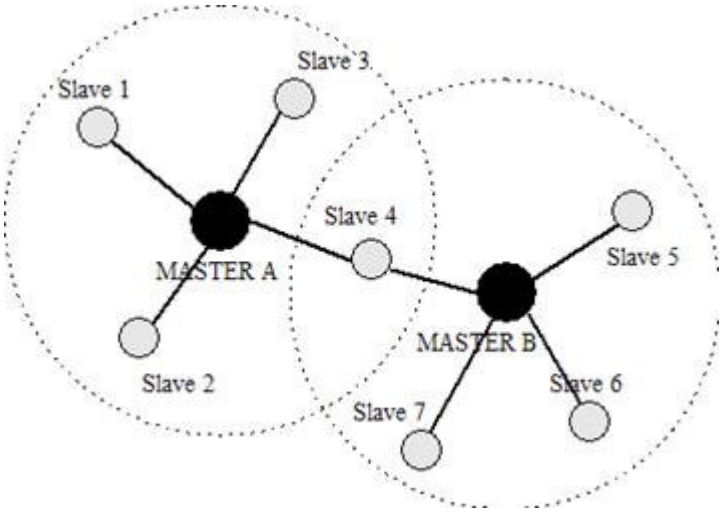

Figure 5. Two piconets connected to a form of scatternet

The Bluetooth core specification version 1.1 [Sig01a] defines a radio link with a maximum capacity for asymmetric data transfer of 723,2 kb/s and 432,6 kb/s for symmetric data transfer. The maximum operational range of this radio link is approximately either 10 meters or 100 meters with power amplifier. Bluetooth devices are capable of both voice and data transmission. The operational frequency is in the unlicensed frequency band at 2.4 GHz to 2.48 GHz. After the core specifications the Bluetooth SIG has produced a new version for Bluetooth specifications [Sig04]. The version 2.0 supports 1 Mb/s bit rate and with enhanced data rate the gross air bit rates are 2 or 3 Mb/s. Version 2.0 operates still in the same frequency channel as the initial core specification has defined.

One Bluetooth module can support up to three simultaneous voice channels, each providing a 64 kbps synchronous voice channel. For data transfer there are two options – asynchronous or synchronous mode. A single Bluetooth device can have up to seven active connections with other Bluetooth devices. If a device functions as a master for several asynchronous connections, it shares the asynchronous data bandwidth among all the active connections. Connections – or rather, devices that are functioning as slaves in a connection – can be parked, i.e. put "on hold" and continue with the data transfer after a while. This way an unlimited number of devices can be virtually connected to a single device but this affects on data transfer rates if many connections operate at the same time.

2.3.2.1 Bluetooth protocol stack

The protocol stack of Bluetooth does not directly follow the OSI model or 802.11 model. The IEEE has however been working on modifying Bluetooth so that it would fit into OSI model better. The Figure 6 shows the Bluetooth protocol stack as it is presented in the [Tan03].

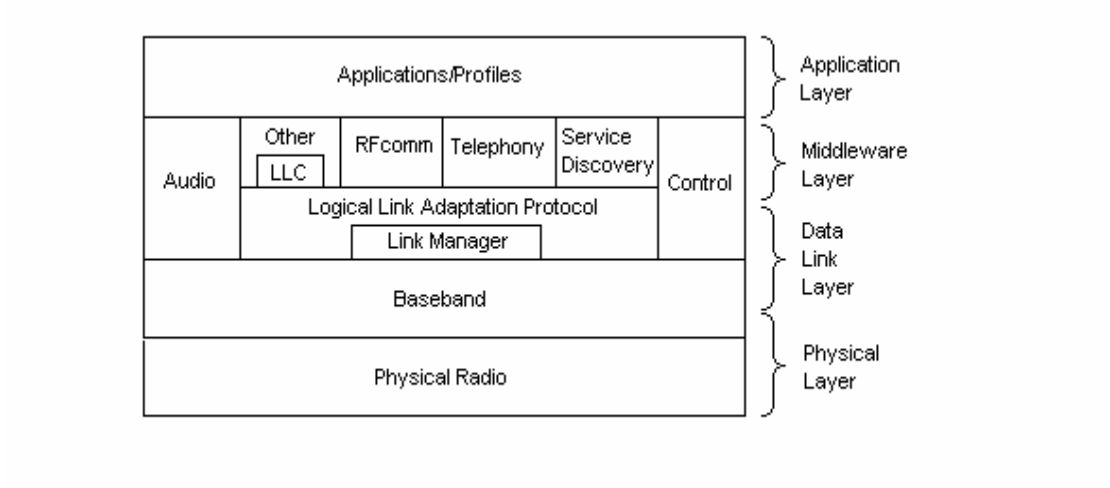

Figure 6. The Bluetooth protocol stack

Physical radio layer defines the used channels, modulation and other aspects that deal with the transferring of bits. As earlier mentioned, the Bluetooth operates in the 2,4 GHz ISM band on the 79 channels, which are the same that 802.11 standard uses. This causes these the two systems to interfere with each other. There are mechanisms for avoiding the interference with other Bluetooth devices, but there is not yet any solution for avoiding interference with the 802.11 devices that use the same frequency. However the IEEE is currently looking a solution to that also. Modulation used in physical layer is frequency shift keying. For fair channel allocation frequency hopping spread is used with 1600 hops/s and a dwell time of 625 µs.

Baseband layer turns raw bit streams into frames and defines some key formats. Each frame is transmitted between master and slave over a logical channel, which the logical link layer handles. There are two kinds of links, ACL (Asynchronous Connection-Less) link and SCO (Synchronous Connection Oriented) link.

The ACL is used when transferring packet-switched data that is available on irregular intervals. This data comes from the logical link control adaptation protocol and it is delivered to the same protocol on the receiving side. Traffic in ACL is delivered on a best-effort basis, meaning that there is no QoS at baseband layer with ACL. Frames can also be lost and have to be retransmitted. A slave can have only one ACL link to its master.

The SCO link is used for real-time data, such as voice transmission. In SCO the channel is allocated on a fixed slot in each direction. Due the time-critical nature of SCO links, the frames are not retransmitted. Instead, forward error correction can be used to provide high reliability. Forward error correction means that each bit to be transmitted is simply repeated two or three times, depending on the type of sent frame. A slave may have up three SCO links to its master.

Logical link control adaptation protocol is called L2CAP and it has three major functions. It accepts packets from upper layers and breaks them into frames that it relays to baseband layer. The size of packets arriving to L2CAP can be maximum of 64 KB. At the receiving side, L2CAP assembles the arriving frames and relays them to upper layers. L2CAP handles multiplexing and demultiplexing of multiple packet sources. It also makes decision for which upper layer protocol arrived and reassembled packets are to be sent. QoS requirements are also handled by L2CAP, both when links are established and during normal operations. Maximum packet size allowed is negotiated at the setup to prevent a large-packet device from drowning a small-packet device. This feature is needed because not all devices can support the maximum sized packets.

Middleware layer contains a collection of different protocols. LLC is the addition of IEEE, which is used to create a compatibility with 802 networks. The RFcomm (Radio Frequency communication) is a protocol that emulates the standard serial port found in PCs. It is based on [TS101.369] specification and is used for connecting devices, like the keyboard, mouse or modem. The telephony protocol is a real-time protocol that is used for call control signalling for the establishment of speech and data calls between Bluetooth devices. Service discovery protocol (SDP) is used in the Bluetooth environment to make service discovery easier but it does not define methods for accessing the services. Each Bluetooth device has SDP service records that define, which services the device supports.

Application layer in Bluetooth standard supports multiple different profiles. The purpose of profiles is to describe how implementations of user models are to be accomplished and they describe a number of user scenarios where Bluetooth performs the radio transmission. The UMA standard refers to the PAN profile when using Bluetooth as an access method.

2.3.2.2 PAN profile

PAN profile expands point-to-multipoint capabilities defined by the Bluetooth 1.1 core specifications making it possible to form ad-hoc networks and providing slave-to-slave communication on top of the 1.1 core specifications. The PAN profile defines three different roles for PAN devices, which provide different services for other PAN devices. The PAN profile also describes how the BNEP (Bluetooth Network Encapsulation Protocol) is used to provide IP connectivity between PAN enabled devices. BNEP generally provides an Ethernet type of networking framework for the Bluetooth networking. The following Figure 7 shows the placement of BNEP in the Bluetooth protocol stack.

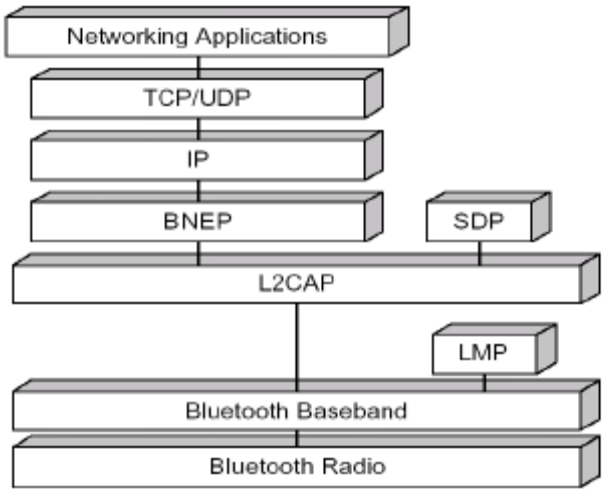

Figure 7. The BNEP protocol in the Bluetooth protocol stack

The UMA standard defines the used roles of a user and an access point. These are PANU (Personal Area Network User) and NAP (Network Access Point).

Network Access Point is a Bluetooth connection capable device that provides some of the features of an Ethernet bridge to support network services. Usually the device with the NAP service forwards Ethernet packets between each of the connected Bluetooth device, but in case of UMA the counterpart may be mobile user with a device supporting only traditional GERAN access method to GSM network. In this case the NAP has an additional network connection to a different network media in which the Ethernet packets are either exchanged via layer 2 Bridging or via layer 3 routing mechanism [Sig01c].

Personal Area Network User is simply a Bluetooth device that that uses either NAP or GN (Group Ad-Hoc Network) service. Specifications of UMA only mention NAP to be used. The specifications of Bluetooth PAN profile give a more detailed description of both NAP and PANU.

2.3.2.3 Comparison between 802.11 and Bluetooth

Both access technologies have their own benefits. It's up to the manufacturers to decide what characteristics they emphasize in access technologies. The table 2 presents comparison of Bluetooth and WLAN technologies.

Table 2, comparison between Bluetooth and 802.11 standards [Sig0a], [Ora00]
* With Bluetooth 2.0 + enhanced data rates

Technology	Bandwidth	Devices per AP	Circuit cost	Power consumption	Range
Bluetooth	3 Mbps*	8 active, hundreds of parked	~5,00$ in 2004	Low (~60mA)	~10m, ~100m with high-powered devices
WLAN (802.11x)	11 / 54 Mbps	~10-30 per access point	~35,00$ in 2004	High (~300mA)	~100m indoors, ~300m outdoors

As the following table shows, the bandwidth is greater with WLAN access methods. Also there can be more devices per access point. But the benefits of Bluetooth are lower power consumption and lower cost in chips. Power consumption can be a very important aspect,

as the battery in handheld devices isn't very powerful. Also the cost of the circuit reflects directly to the cost of the end-product, which can have influence on the consumers.

2.3.3 Using Access Points

Discovering of access points in UMA is based on predetermined procedures. The UMA standard makes a recommendation that the access point should periodically send a periodic beacon that the mobile stations monitor. By that the mobile stations in the service area are able to locate the access point. The standard recommends that the beacon signal is sent at least every 100ms interval.

UMA supporting devices have two operating modes, which are the GSM mode and the UMA mode. At any time the device is in one of those modes. User can affect to this mode selection by choosing a preferred mode type from the device. These modes are:

- GERAN only, where only GSM mode is used.

- GERAN preferred, where the MS uses GSM in cases where it's possible. If the MS detects a bad connection or the connection is lost to GERAN, then the handover procedures to UMAN are started.

- UMAN-preferred, where at any time when the MS is in GSM mode, it can handover to UMAN if UMA coverage is available. However this cannot be done if the MS does have the PLMN search in progress.

- UMAN-only, where after the initial power-up sequence to GSM mode, the MS will never switch to GSM mode.

The access point is identified by its MAC address. The MS obtains this address via the broadcast from the AP and sends it to the UNC when the MS requests UMA service. This identification may be used by the UNC to support location-based services. It can also be used by the service provider to restrict UMA service access in case of unauthorized access points.

2.4 The UMA Network Controller

UMA network controller handles the connection between the GSM core network and broadband IP network. It provides functions equal to GERAN base station controller. UNC also includes a Security Gateway (SGW). SGW handles the security issues, like authentication and encryption between the MS and the UNC. Generally the functions that UNC must support are [UMAa05]:

- Up user plane speech services. Transporting of voice is handled between the Up- and the A-interface.

- Up user plane data services. Transporting of data is handled between the Up- and the Gb-interface.

- SGW to terminate secure remote access tunnels from the MS

- Up control functionality, like registration for UMA services and setting up bearer paths for circuit- or packet-switched services.

- Transparent transfer of upper layer messages between the MS and core network.

UNC is identified either with a Fully Qualified Domain Name (FQDN) or with an IP address. IP address means the Internet Protocol Address that is a unique number consisting of 4 octet-parts separated by dots. FQDN is another type of address with the combination of the host name and the domain name so that the host name is presented first. Example of FQDN would be www.tietoenator.com, where www is the host, tietoenator is a second-level domain and com is the top-level domain. The UMA specification [UMAa05] define the following forms for provisioning UNC and SGW:

- Provisioning UNC: punc.uma.mnc*nnn*.mcc*mmm*.3gppnetwork.org

- Provisioning SGW: sgw.uma.mnc*nnn*.mcc*mmm*.3gppnetwork.org

In previous forms, the *nnn* is the MCC (Mobile Country Code) part and *mmm* is the MNC (Mobile Network Code) part of the International Mobile Subscriber Identity (IMSI) number associated to the subscriber. MCC has always three digits and the MNC has always two digits. This general format of the phone number makes it possible to use the same subscriber identification in various networks of different countries for example.

2.4.1 Discovering of and registering to UMA network

The Discovery procedure is performed by the MS at its first attempt to obtain a UMA service. It first connects to provisioning UNC and then discovers the default UNC. By this, the MS determines the identity of the default UNC that may also be the serving UNC for that connection. The default UNC is always located in the subscribers HPLMN and this is the UNC that the MS must register first. The discovery procedure requires that the MS has established connection to access point and also that the MS does not have GERAN-only mode activated.

The discovery procedure has two important purposes. Firstly it informs the UNC that a MS is connected using a particular AP and has an IP address associated to it. The UNC keeps this information stored for the purposes of providing services. Secondly it provides the MS with needed operating parameters needed when using the UMA services.

After successful discovery the MS can start to register to the default UNC. First it establishes a secure tunnel to the SGW associated with the default UNC. Then the MS sends a register request message including parameters of cell identity (CID), location area identity (LAI) and IMSI of the mobile subscriber. The default UNC, after receiving the message with provided information, sends a register redirect message, which gives MS the identifiers of the serving UNC. In some cases the serving UNC can also be the default UNC. In this case the register redirect message is not sent, but the UNC accept registration with register accept message. After register redirect the MS connects to the serving UNC and provides the same information as to the default UNC. The serving UNC either accepts or rejects the registration from the MS. Reject can happen for example if the IMSI of the subscriber is not allowed to use UMA services. Other example would be when there is

network congestion in the serving UNC but in this case the serving UNC should provide an alternative UNC address to the MS. [UMAa05]

2.5 UMA specified protocols

Protocols define the format of data and the set of rules to be followed in the data communication and network environments. As in UMA, there are many different protocols to be used in different communication events between different network elements. Many of the used protocols are already defined and standardized, but the UMA standard also define two new protocols, the UMA-RR and the UMA-RLC, which are used when the MS communicates with core network using UMA access methods. Next chapters give some information of the two new protocols and also reviews some other used protocols.

2.5.1 UMA Radio Resource protocol

UMA Radio Resource protocol (UMA-RR) is used instead of GSM-RR when the MS is using UMA as access method for circuit switched signalling and user plane traffic. The UMA-RR protocol is terminated in the UNC, which makes the UMA-RR protocol to interwork with the BSSAP protocol that is used with the communication between the UNC and the MSC. The main functions of the UMA-RR described in [UMAa2005] are:

- Handling registration procedures between the MS and the UNC.
- Setup a bearer path for circuit switched traffic between the MS and the UNC.
- Handover procedures between GERAN and UMAN.
- Functions for GPRS supervision methods, for e.g. call suspension, paging, ciphering and configuration.
- Support for identification of an access point being used with UMA access method.

UMA-RR has functions very similar to GSM-RR. In GSM network the radio resource management has functions that are used to establish and release stable connections between mobile stations and an MSC for the duration of a call and maintain them despite user movements. Radio resource management protocol must also cope with a limited radio resource and share it dynamically between all needs.

Figure 8 shows the protocol stack of UMA-RR and used interfaces for signalling and user-plane traffic in circuit switched case.

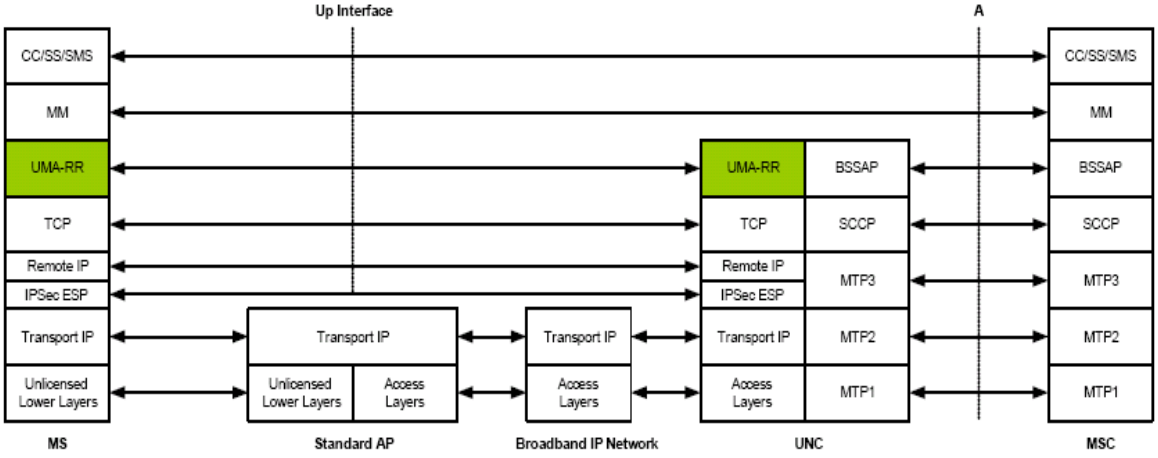

Figure 8. The protocol stack of UMA-RR in signalling [UMAa05]

As the Figure 8 shows, the upper layer protocols of mobility management (MM), call control (CC), supplementary services (SS) and short message service (SMS) are carried transparently between the MS and the MSC, as they don't affect on UMA access methods procedures, signalling or user plane traffic directly.

Mobility management (MM) has two main purposes. As the name implies, the first is for handling the mobility of the mobile station. The second purpose concerns with security management issues.

Mobility of users and devices is one of the most important aspects when we're dealing with mobile communications. Otherwise it wouldn't be any real mobile communication, if the devices could only be used in some fixed locations. The ability to use a mobile device

in any place, where there is network coverage, is a huge benefit, but it also makes things bit more complicated, as the network elements must be able to handle the mobility correctly.

In fixed networks, each subscriber is connected to one local switch, usually for a long time. Every call involving this subscriber goes through this switch. It makes things simpler, as there is only one place needed to hold the subscriber related information. However, this does not apply in mobile communications as the subscriber can be in any location that has network coverage.

The mobility management protocol with UMA works in a similar way as in GSM network. The features of MM include location management, roaming, system load control and failure recovery procedures. More detailed description of aforementioned features can be found in various sources, for example from [Mouly92].

Call control (CC), **Supplementary services** (SS) and Short **Message Services** (SMS) belong to **Communication Management layer** (CM). The functions of CM is to set up connections between users, maintaining those connections and also releasing them. Connections can be normal calls with voice or data transfer connections or any other that is supported by lower levels.

2.5.2 UMA Radio Link Control

UMA-RLC is used to handle the GPRS signalling. In UMA, the GPRS-RLC is replaced with UMA-RLC, which has equivalent purposes. However there are differences, as with the transport characteristics of Up interface, the GPRS Temporary Block Flow (GPRS TBF) cannot be used. The TBF is a connection between the mobile device and the network and it allows point-to-point transfer of data. One of the main purposes of GPRS TBF is also to ensure the transfer of packets to correct devices. In each TBF, the packets are labelled with Temporary Flow Identifiers (TFI), which are used to identify packets and to associate the mobile device with the current TBF. But as mentioned before, the UMA-RLC doesn't use these methods for transferring of packets. Instead, reliability in UMA is

accomplished with the usage of TCP. This results in that the UMA-RLC is much lighter than the GRPS-RLC.

Also like in UMA-RR, the UMA-RLC is terminated in the UNC, so the usage doesn't require any changes into existing core network. The main functions of UMA-RLC described in [UMAa2005] are:

- Delivery of GPRS signalling and SMS messages over a secure tunnel
- Paging, flow control and GPRS transport channel management
- Transfer of GPRS user plane data

The Figure 9 shows the protocol stack of UMA-RLC and the used interfaces and network elements.

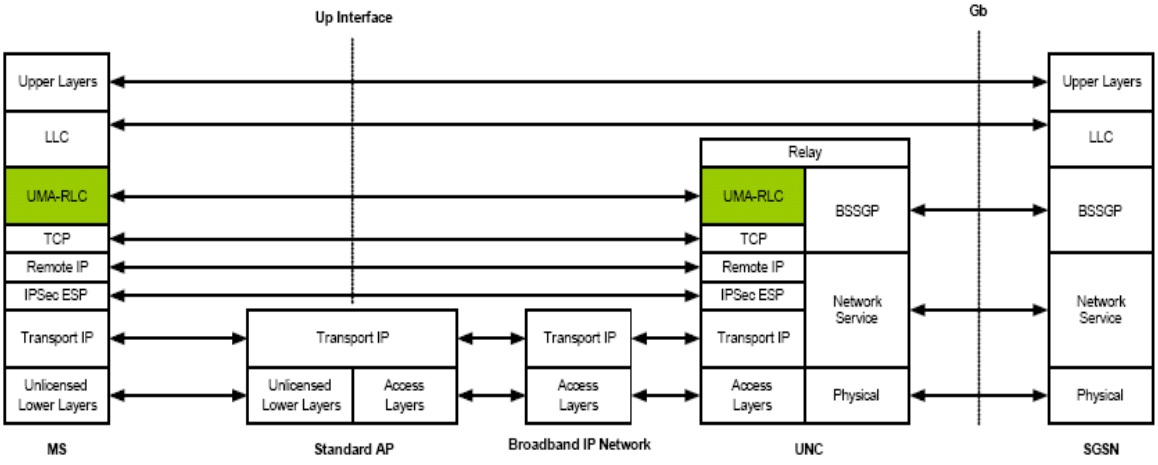

Figure 9. The protocol stack of UMA-RLC [UMAp2005]

As the figure shows, upper layer protocols are carried transparently between the MS and the UNC. LLC is described in previous chapter along with 802.11 and Bluetooth.

2.6 Security in UMA

As the UMA uses unlicensed radio access points and external broadband IP network, the security issues are a very important aspect. The UMA standard defines several security mechanisms that are used to provide authentication, encryption and integrity. The security mechanisms that are used are common and standardized mechanisms that are applied with the UMA standard, e.g. IPSec is used to protect the packet transfer between the MS and the UNC. There can be some variation with the used profiles of IPSec and IKEv2 in MS.

The Security Gateway (SGW) is a network element that is used for terminating secure tunnels from the MS, providing mutual authentication, encryption and data integrity for voice, data and signalling traffic.

Security mechanisms can be divided into four levels depending on where they apply. The second level is the most important level as it is directly connected to UMA specifications. Other levels have security methods that are already defined as they are commonly used and implemented technologies. Figure 10 shows the four levels and where they apply in the UMA standard and its elements.

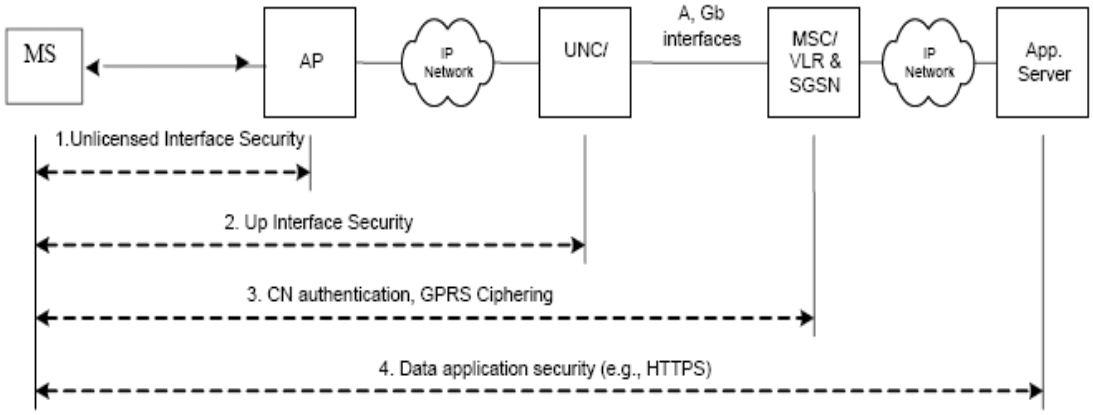

Figure 10. The security mechanisms in different levels

The first level is between the MS and the AP, where the unlicensed radio access interface is used. Security mechanisms apply to voice, data and signalling over the radio interface. These mechanisms are dependant on the used access technology. The standard doesn't

define any mandatory security mechanisms for the Bluetooth or 802.11, but makes recommendations to use them. In 802.11 the MS should support WEP with WPA and PSK for authentication. The WEP (RC4) and WPA (TKIP) should be used for encryption. In Bluetooth access methods, the standard makes a recommendation just to use Bluetooth security but it is optional. The more detailed description of Bluetooth security methods can be found from [Sig01b] and the description of 802.11 security methods from the IEEE standards for different 802.11 versions.

The second level is between the MS and the UNC and it applies to the Up-interface. Specifications define the following methods of security:

For authentication mechanism the mutual authentication is accomplished by using either EAP-SIM or EAP-AKA and IKEv2. These methods are described in later chapters. For confidentiality and integrity of data the IPSec ESP method is used. Security in second level defines that before mentioned security methods apply to all traffic (user-plane and signalling) between the MS and the UNC.

Third level is between the MS and the core network. Authentication of the subscriber is between the MSC/HLR or SGSN and the MS and it is transparent the UNC. There is however a cryptographic binding between the MS-CN and MS-UNC authentication to prevent the man-in-the-middle attacks. GPRS ciphering is the standard ciphering used with GPRS LLC. More detailed information can be found at [TS43.020].

Fourth level is the additional application level security and can be applied in end-to-end communication of the MS and the application server or gateway. Methods for these can be e.g. SSL session over HTTP protocol.

2.6.1 IPSec

As the IP traffic is generally unsecured, there have to be used some methods to provide the needed security. In UMA, the IP Security architecture, shortly as IPSec, is a protocol suite that is used for providing that security for IP connection between the MS and the UNC. IPSec protocol family was developed by the IETF and currently there are over ten RFCs

that deal with the aspects and methods of IPSec. The RFC 2401 - Security Architecture for the Internet Protocol, serves as the basis for IPSec protocol suite.

Main purpose of IPSec is to define the provision of various security services for traffic at the network layer for both IPv4 and IPv6 thus providing protection of transparent nature of upper layer protocols. The security service include the following:

- Access control
- Connectionless integrity
- Data origin authentication
- Protection against replays
- Confidentiality (encryption)
- Limited traffic flow confidentiality

IP protocol disassembles the transported data into packets that are routed to destination (or destinations in some cases). Packets contain a header field and a data field. The header field contains among other fields a source IP address as well as a checksum that allows the receiver to detect whether the packet has been corrupted during the transmission. This checksum is a cyclic redundancy check only and does not offer any defence against intentional modifications of the datagram. Also the IP address in the header field is not necessarily the address where the datagram was sent. This can be exploited in source routing attacks where the attacker modifies the IP address in sent datagrams.

2.6.1.1 Authentication Header and Encapsulating Security Payload

IPSec provides two main methods for protecting the integrity of transferred packets. These methods are:

- IP authentication header (AH), detailed in RFC 2402
- IP encapsulating security payload (ESP), detailed in RFC 2406

The IP authentication header protects the integrity and authenticity of IP datagram but does not provide confidentiality. Authentication data is placed in a header within the datagram. The Figure 11 shows the structure of authentication header.

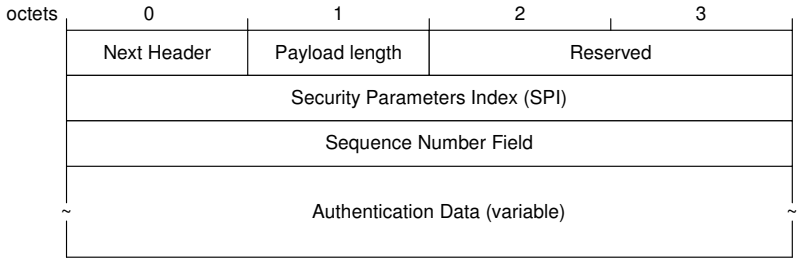

Figure 11. Authentication Header structure [RFC2402]

- **Next Header**. Contains the type of the payload which follows the AH header. For instance, if TCP follows the AH header, the value is 6.
- **Payload length**. Contains the length of the AH header in four octet chunks, not counting the first eight octets.
- **Reserved.** For future use.
- **Security Parameters Index (SPI)**. Used to identify the corresponding SA.
- **Sequence Number**. An increasing counter value assigned by the sender. The receiver may use this field to recognize replayed packets and discard them.
- **Authentication Data**. A cryptographic integrity check value (MAC) of the data.

The placement of authentication header in the IP packet is shown in figure 12.

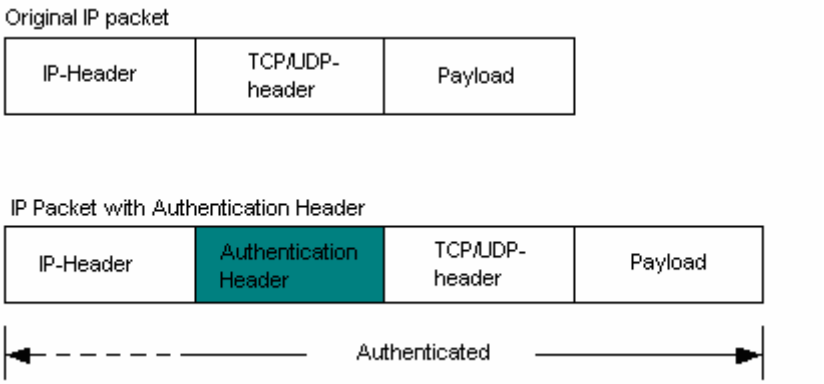

Figure 12, IP packet with Authentication Header

To authenticate a datagram, the sender must first locate a security association (SA), specifying parameters like the integrity check algorithm, the cryptographic key and the size of the authentication data. Normally user identity, destination address and security parameters index determine which SA will be used. The receiver of the datagram checks the SPI and destination address and locates the relevant SA and verifies the authentication data. If authentication fails, the receiver logs the failure and discards the packet.

The Encapsulating Security Payload (ESP) is used to protect the confidentiality of IP packets. Depending on the encryption algorithm used, ESPs may also protect integrity and authenticity. ESP can also be used to detect multiplied datagrams and prevent their malicious usage. ESP is also put into header of a datagram and the structure of ESP is presented in the following Figure 13.

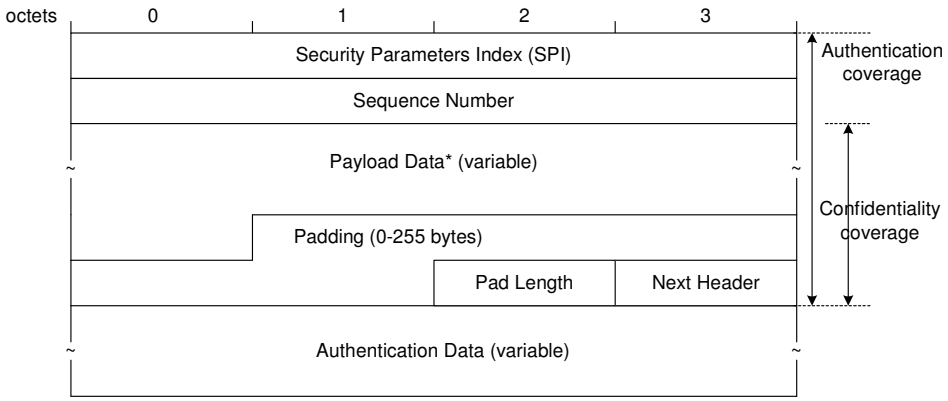

Figure 13. Encapsulating Payload Structure [RFC2406]

- **Security Parameters Index (SPI)**. Used to identify the corresponding SA.
- **Sequence Number**. An increasing counter value assigned by the sender. The receiver may use this field to recognize replayed packets and discard them.
- **Payload Data**. Contains the IP payload protected by ESP. If the algorithm used to encrypt the payload requires the Initialization Vector (IV), it is located in this field, but is not encrypted.

- **Padding**. Padding is used for several purposes: To align the fields from the Payload Data to the Next Header to a multiple of four octets, to align the size of Payload Data suitable for the cryptographic algorithm and to increase the size of Payload Data to hide the actual length of the plaintext payload (limited traffic flow confidentiality).

- **Pad length**. Number of octets of padding.

- **Next header**. Identifies the type of data contained in the Payload Data field, e.g. the upper layer protocol identifier, like TCP.

- **Authentication Data**. Contains the authentication value computed over the ESP packet excluding this field. A cryptographic integrity check (MAC) value of the data. This field is present, if the SA specifies the ESP with authentication.

Before encrypting a datagram, the sender locates a security association to determine which encryption algorithm and key to use. Security association in AH and ESP are not the same. After locating SA the sender can choose from two different ESP modes:

- The transport mode, an upper layer protocol frame, e.g. TCP or UDP, is encapsulated within the ESP and the IP header is not encrypted. This transport mode provides end-to-end protection of datagram transfer between two nodes.

- The tunnel mode, where an entire IP datagram is encapsulated within the ESP, which is then transmitted within another IP datagram. After sender has chosen ESP mode and transmitted the packet, the receiver of the datagram locates the relevant SA and decrypts the encrypted payload. If decryption fails, the failure is logged and the datagram is discarded.

2.6.1.2 Internet Key Exchange Version 2

Internet Key Exchange (IKE) is a general-purpose security exchange protocol, which provides mechanisms for generating authenticated keying material for the negotiating

peers. IKE is a component of IPSec and is used for performing mutual authentication and establishing and maintaining security associations. It was initially defined in RFCs 2407, 2408 and 2409, but the version 2 defined by the IETF makes those RFC obsolete.

IKEv2 has the following changes compared to its previous version [Ala03]:

- The whole protocol is defined in one document instead of three. This includes NAT-compatibility, extended authentication and extension defining of remote address.
- The protocol is simplified by replacing all methods used in first phase mode with one method.
- Useless and confusing fields from the header are removed.
- The protocol is made faster by reducing the initial message transaction to four.
- The cryptographic syntax protecting the IKE messages is changed with one based closely on ESP to simplify implementation and security analysis.
- The number of possible error states is reduced by making the protocol reliable (all messages are acknowledged) and sequenced.
- The stability of the protocol is increased by allowing the responder to not do significant processing until is receives a message proving that the initiator can receive messages as its claimed IP address.
- Cryptographic weaknesses such as problem with symmetries in hashes are fixed.
- Traffic selectors are specified in their own payloads rather than overloading IP payloads.
- Recovery from certain errors is specified in order to make it easier to make future revisions in a way that does not break backward compatibility.
- Shared state maintenance in the presence of network failures and denial of service attacks is simplified.

The specification of IKEv2 contains a lot of options for the cryptographic algorithms. For UMA, the IKEv2 is profiled so that the implementation requirements can be limited. The specifications of UMA define four cryptographic suites that are applied. The UNC must

support all of those suites and the MS must support at least one of them. Detailed information of the used suites can be found at UMA specifications [UMAa05].

2.6.2 Extensible Authentication Protocol

Extensible Authentication Protocol is a mechanism for authentication and session key distribution using the Global System for Mobile Communications (GSM) Subscriber Mobile Identity (SIM). Haverinen et al describe EAP-SIM as follows in their specification document [Hav03a]: "*The mechanism specifies enhancements to GSM authentication and key agreement whereby multiple authentication triplets can be combined to create authentication responses and session keys of greater strength than the individual GSM triplets. The mechanism also includes network authentication, user anonymity support and a re-authentication procedure*"

EAP-AKA is a similar protocol to EAP-SIM. It is based on the mechanisms of authentication and key agreement mechanism, specified by the 3GPP for UMTS in [TS33.102] and for CDMA2000 in [S.S0055-A]. Compared to EAP-SIM, EAP-AKA has some more useful benefits, which include the following [Hav03b]:

- AKA can also be used as a secure PPP authentication method in devices that already contain as identity module
- The possibility to use 3rd generation mobile network authentication infrastructure in the context or wireless LANs.
- Relaying on AKA and the existing technology in a seamless way with any other technology that uses EAP.

Both EAP-SIM and EAP-AKA are supported in UMA. It depends on the capabilities of MS which method is used. The MS with SIM only or MS with USIM but not capable of UMTS AKA, the authentication is performed using EAP-SIM inside IKEv2. If the MS is has USIM and is capable of UMTS AKA, the EAP-AKA authentication method is used within IKEv2.

With UMA, the EAP-SIM and EAP-AKA procedures are performed between MS and AAA server. After the MS has connected to AP and found appropriate UNC, it initiates the connection with UNC by starting the IKEv2 initial exchanges. As a result of these exchanges, the EAP-SIM or EAP-AKA procedure is started. The UNC-SGW acts as relay for EAP-SIM and EAP-AKA messages. After the EAP procedure has been completed, the IKEv2 procedure can be continued as the signalling channel between MS and UNC-SGW is secured. The MS and UMAN can then continue with the discovery or registration procedures. The UMA specifications have more information about the message exchange in both EAP-SIM and EAP-AKA procedures [UMAa05].

3. TESTING

The following chapters concentrate on the common issues of testing and how the testing of new technologies, like UMA, can be achieved in certain defined areas. The idea is to present general aspects of testing theory and also different testing methods. A separate chapter is reserved for protocol testing, as it is an essential part of this thesis work. Fourth chapter then presents a testing implementation that is used to test the UMA specified protocol.

Importance of testing cannot be argued. Some claims are made, that the testing can account for more than 50% of a lifetime cost of software application or system [Beiz90 p.1]. That being said, correctly used testing methods can really affect the cost of software engineering in general. Well-planned testing scenarios can save a lot of time and money and can also result in better quality of software products. Also without testing, there is no practical evidence that the software system is free of errors, and that it works as specified.

Today's software applications and systems have grown to be rather large and complex. Many different blocks interoperate with each other in such a way that it is hard to predict to whole operability in the same phase as the implementation goes on. Of course good planning on implementation helps a lot, but the time spent on planning is always away from something else. Software business nowadays puts a lot of pressure on rapid development of new products, as the markets are very time-critical. Hectic business combined with large and complex systems can result in poor software quality with lots of errors, bugs or other unwanted behaviour.

To ensure certain quality and correct operability in software products, the developers of software need to use different testing methods. What comes in place is a well define testing plan, that identifies the areas where testing should be done and how it should be done.

Testing methods have also improved greatly from the beginning of the days. Automated testing tools, more effective methods and other efforts put on testing theory evolutions have made it possible to create software with higher quality in less time.

3.1 Testing methods and aspects

Definitions of testing have been varying from time to time. The Table 3 gathers some thoughts that have been used for defining the meaning of testing over the years.

Table 3. Definitions of testing [Cra02]

Year	Definition
1979	Testing is the process of executing a program or system with the intent of finding errors.
1983	Testing is any activity aimed at evaluating an attribute of a program or system. Testing is a measurement of software quality.
2002	Testing is a concurrent lifecycle process of engineering, using and maintaining testware in order to measure and improve the quality of the software being tested.

The exact methods and techniques of software testing can be different in each situation, but the main objective is always the same: to reduce the level of uncertainty about the quality of a system. There are many different ways to define software quality. Quality can be viewed as innate excellence of a piece of software, and it can be measured, for example, in terms of product characteristics, user satisfaction, and acceptable price. For the purpose of software testing, quality is usually equated with conformance to stated requirements. Therefore, any deviation from specified requirements or from international standards is seen as a reduction in quality.

3.1.1 Different levels of testing

Software testing in general has three main levels: unit / component testing, integration testing and system testing. The objectives of testing each level are different and therefore in testing processes the relative portion of tests with different levels vary on tested software and environment. Also the boundaries between levels are vacillating. Still there exist some generalizations for each level of testing.

Unit testing concentrates on testing of the smallest parts possible to test in software. Usually they consist of several dozen to several hundred lines of code and are work of one programmer. The idea of unit testing is to show that the unit doesn't satisfy its functional specification and / or implementation structure doesn't match the intended design structure. When faults are discovered, they are referred as "unit bugs".

Component testing deals with components, that are integrated aggregates of one or more units. In component testing, the goal is to show that the component doesn't satisfy its functional specification and / or implementation structure doesn't match the intended design structure. Errors discovered in component testing are referred as "component bugs". [Beiz90]

Integration testing is used for exposing problems that can occur when components are combined. Even if a component works properly individually, it may not work properly when used in system with other components. Examples of inconsistencies between components are improper call or return sequences, inconsistent data validation criteria and inconsistent handling of data objects. [Beiz90]

System test concentrates on revealing bugs that cannot be attributed to components as such but are rather inconsistencies between components or inconsistencies in planned interaction of components and other objects. System testing concentrates on higher-level methods including testing for performance, security, accountability, configuration sensitivity, start-up and recovery. [Beiz90]

3.1.2 The V-model of testing

Different testing levels described in previous chapters correspond to actual testing process according to V-model. In this model the testing is planned on and done in several levels. The Figure 14 illustrates how the planning of testing is directly linked to the planning and documentation of the system. The idea is to execute the testing procedures in reversed order as they are planned. The acceptance testing, that is planned first, is executed last and the unit testing, that is planned last, is executed first.

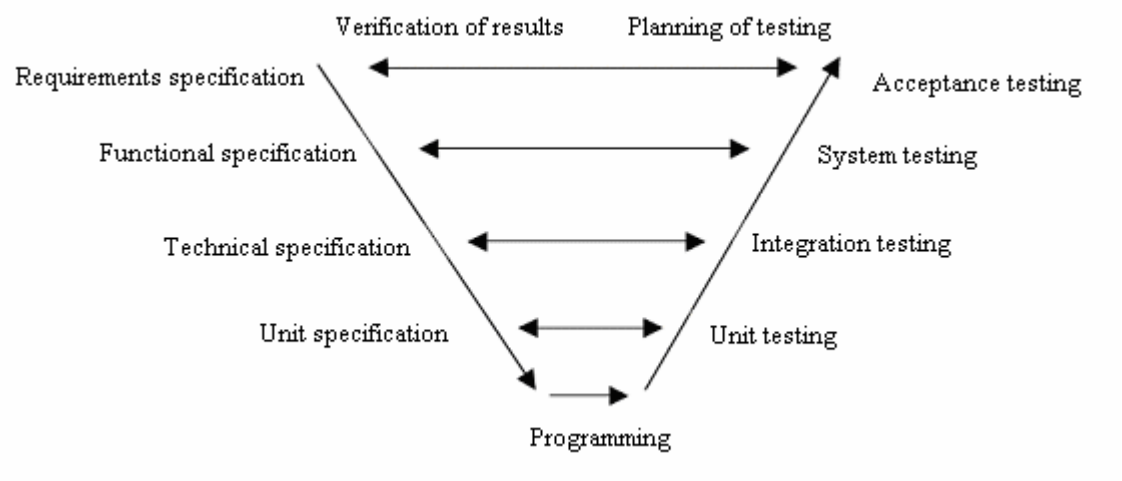

Figure 14. The V-model of testing [Hai01]

The acceptance testing is a testing that validates the system against to the end-user requirements. This is usually done to ready system that has completed other testing phases. The V-model strongly emphasizes the importance of testing as the testing plan is constructed right from the beginning of a software development process. Each level of testing can be planned right after the corresponding specification is written.

3.1.3 Test planning

"*The plan is nothing, the planning is everything.*"
-Dwight D. Eisenhower

Good test planning is one important key element for any successful testing procedure. Test planning is a process that ultimately leads to a document that allows all parties involved in the testing process to decide what the important issues are and how to best deal with these issues. The goal of test planning is not to create a long list of test cases, but rather deal with the issues of testing strategy, resource utilization, responsibilities, risks and priorities.

Creating a test plan for a specific level requires a clear understanding of the unique considerations associated with that level. But before that, it is recommended to create a Master Test Plan (MTP), which purpose is to orchestrate testing at all levels. These levels in MTP correspond the levels described in chapter 3.1.1. In addition to MTP, it is necessary to create detailed or level-specific test plans. With larger projects it is worthwhile to create at least plans for four previously described levels. In smaller projects it may be possible to have just one test plan that covers all the levels of test. [Cra02]

Test planning should begin as soon as possible. Generally it is good to begin the MTP at about the same time as requirements specifications and the project plan are developed. This can have a significant influence on the content of the project plan as it may clarify some important aspects about the project itself. The Figure 15 presents the common guidelines for timing of the test planning in different levels.

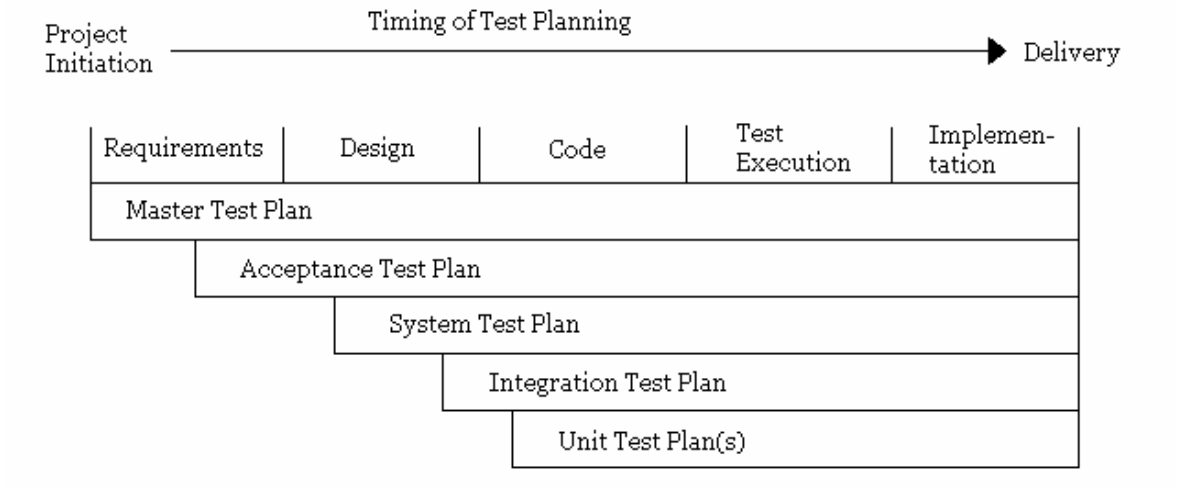

Figure 15. Timing of Test Planning [Cra02]

As for detailed information of the actual test plans and how to construct them, there are many different versions and methods available from many different sources. Even the IEEE standardization organization has developed a standard template for test planning [Std829], which identifies many aspects that need to take account when the test plan is constructed. Also different books and other sources mention their own methods for creating a test plan and what kind of plan it should be. Although they have somewhat different methods and plans, they all comply quite much of the IEEE test plan standard

template. The following method of test planning steps is taken as an example and it can be found at [Roy93].

Test plan steps

1. Identify the requirements
2. Associate requirements to software units
3. Establish a verification sequence
4. Identify the software aggregates required for requirement verification
5. Establish a sequence diagram
6. Identify supporting material
7. Attach dates and durations
8. Iterate
9. Partition the system into blocks and builds
10. Write the plan

3.1.4. Black-box testing

Black-box testing is sometimes referred also as functional testing. With this approach the system under the test is seen as a “black box”, which takes some specified inputs and produces some outputs. These outputs are then verified for conformance of specified behaviour. The internal structure of software is not concerned at all. System testing is a good example of black-box testing. System tests are often based solely on functional specifications, and information on the internal structure of the software program under testing is not used at all.

3.1.4. White-box testing

White-box testing, which sometimes is called also as glass box testing, is a testing method that concentrates on examining the inner structure of the programs code. The goal is to systematically cover all parts of the program, e.g. statements, branches etc. The coverage percentage is measured with different methods and tools and the knowledge of the code is used when designing the test cases. Unit testing is an example of white-box testing, as in unit testing it is important to know the internal structure of the tested unit.

3.1.4. Hybrid methods

Since it is not practical to carry out testing using only pure black-box or white-box methods, it is a good solution to blend those two methods. The exact proportion of black-box and white-box methods depends considerably on the software development environment. In general, the less the testers know about the environment in which the software component or program operate, the more the testing strategy will lean toward white-box testing. Conversely, the less the testers know about the construction of a component, the more it's likely to be tested using black-box method.

In software development, the mix of black-box and white-box methods will usually vary with time. During early development and program integration, when components are being examined more or less in isolation, the emphasis will likely be more on the white-box approach. When entire functions become available, and when the whole system nears completion, testing will shift to black-box way of testing. This applies to V-model so that usually unit and component testing have a white-box approach to testing as the system and acceptance testing are carried out with black-box methods.

3.2 Protocol testing

A mobile terminal is a software intensive product, and protocol software is a critical part of any generation mobile terminal. This applies to UMA supporting mobile stations and network elements as well. But before we discuss more about the protocol testing, it is good to define what is a protocol and what it does.

According to [Tan03] a protocol can be defined as follows: " A protocol is an agreement between the communicating parties on how communication is to proceed". Protocols determine how data is transmitted between computing devices and over networks. They define issues such as error control and data compression methods.

Although protocol testing follows mainly the same methods as normal software testing, it has some minor discrepancies. The Figure 16 shows the basic structure of generally used protocol testing methods.

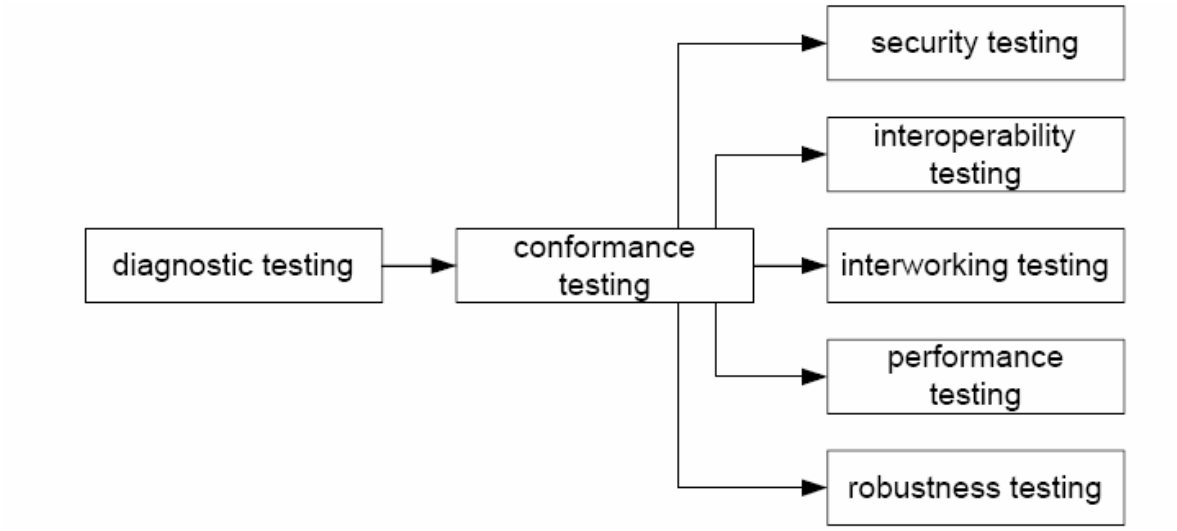

Figure 16. Protocol testing methods [Tak00]

Diagnostic testing is a testing of protocol implementation with the complete access to internal structure of implementation. It can also be seen as white-box testing. The goal of diagnostic testing is to ensure that the protocol is bug-free and operates correctly. However, the diagnostic testing doesn't guarantee the conformance of a protocol. It is normally performed on new protocols, which are not yet been conformance tested.

Conformance testing is a testing of protocol implementation against the protocol specification. It has been the most studied type of protocol testing, much because of a standardized framework and methodology exists, as defined in ISO/IEC 9646 standard. The intent of the protocol standards is to specify open interfaces and interworking with products of different suppliers.

Conformance testing is carried out by using black-box testing approach to verify that the protocol implementation is able to perform a selected subset of all signal exchanges defined in the protocol specifications. Practical limitations make it impossible to test the protocol exhaustively and therefore four types of conformance tests are considered significant [Tak00].

1. Basic interconnection tests, which provide limited testing of the protocol in relation of the main features specified. The purpose is to verify that there is sufficient conformance for interconnection between the test system and protocol under test.
2. Capability tests, which provide limited testing of each of the static conformance requirements in protocol specifications. The purpose is to ascertain which capabilities can be observed and to check that those capabilities are valid with respect to the static conformance requirements.
3. Behaviour tests, which test the protocol implementation as thoroughly as it is practical, over a full range of dynamic conformance requirements specified in protocol specifications.
4. Conformance resolution tests, which probes in depth the conformance of the protocol implementation to particular requirements in order to provide a definite yes/no answer in relation with specific conformance issues.

The implementation part of this thesis work concentrates on the basic conformance test of the UMA-RR protocol. The goal is to verify that the protocol conforms its specifications and also to test the capabilities of the protocol in used testing environment. The testing implementation and aspects related to it are presented in chapter 4.

Security testing is testing or protocol implementation against its security requirements. The purpose of security testing is to reveal vulnerabilities in protocols and make the implementation quality better in order to prevent attacks. Security testing should include both syntax and semantics testing of a protocol. [Mus00]

Interoperability testing is testing of protocol implementation in real environment, compared to conformance testing, which is usually carried out in simulated environment. The goal of interoperability testing is to verify that the protocol interacts with other entities properly in real environment also.

Interworking testing is testing of protocol implementation interworking with other protocols, services and networks. It's much like interoperability testing but the difference is that equipment interoperates while protocol, services and networks interwork. The following Figure 17 explains the difference.

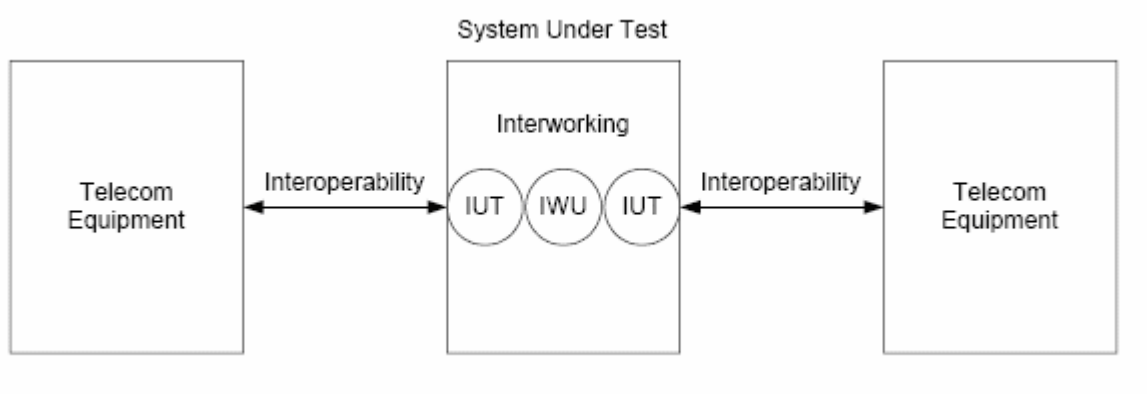

Figure 17. The difference between interoperability and interworking [Tak00]

Performance testing comes in place after software or a protocol has been verified with system testing and its interoperability with integration testing. Sometimes there is no need for performance measurements but especially with communication protocols it is an important issue.

There are usually two different test methods in performance testing. First one is the general stress test, where load balance is usually 70-80%. This simulates “normal” situation where decent amount of load is generated for a longer period of time. The idea is to verify that the system under test is capable of handling longer-term usage with load that is situated to estimate normal utilization rate.

The second method uses peak load, which is the maximum load the protocol can handle meeting all the performance criteria set. The load is increased to a point, in which the offered load and accepted load are both the same. Beyond this point the protocol starts rejecting some of the incoming messages, frames or signals. This type of performance testing method is used to verify that the protocol can accept the predefined engineered load.

The third method targets to check how the system deals with overload situations. With overload testing scenario, the load balance is over 100%, sometimes as high as 150%. This type of testing is usually done with peaks of overload over certain small periods of time rather than with continuous overload over long period of time. The purpose of this

performance testing type is to verify the systems capabilities to handle overload in a way that doesn't cause the system crash or make other larger problematic failures.

Robustness testing is a testing of protocol implementation in an environment behaving erroneously. The goal of robustness testing is to verify that the protocol works properly under abnormal conditions. This testing method is closely tied to security testing, as robustness testing might reveal some errors that reflect directly on security.

4. PROTOCOL TESTING IMPLEMENTATION

In this thesis work, the purpose of this chapter is to present a testing implementation that is used to test the UMA-RR protocol. Testing will concentrate only on protocol software testing and more particularly on testing of conformance of the protocol. This chapter also deals with used testing environment and platform.

Target is set to create a testing implementation that is capable of fulfilling the requirements specified for UMA-RR protocol functionality. This means the usage of correct states, timers and messages defined in UMA protocol specifications. At this point, the main goal for testing is to verify the behaviour of the protocol in testing environment according to protocol specifications. This is important as the future testing is planned to carry out with multiple network elements to simulate interoperability, interworking and performance of the protocol. These aspects are, however, out of scope of this thesis work.

Protocol testing implementation is made with CVOPS. The main reasons for using CVOPS are the portability of the code and easy-to-use implementation framework. Also the protocol implementation with CVOPS is not tied to any specific architecture of system that implements the UMA protocol. CVOPS offers support for C - and C++ - languages for testing implementation. Testing implementation in this thesis work is made with C - language.

Protocol testing in this thesis work concentrates on diagnostics testing and conformance testing as the protocol implementation is tested in simulated environment. At first, when the protocol is constructed, the testing can be seen as white-box testing to verify that the implementation works as intended. This basically means checking of function calls, memory handling, parameter handling etc. in more detailed level. Also initialization phase is examined and verified so that file paths are correct and also that virtual tasks are in correct states and ready to accept service primitives. After this, the protocol is tested for conformance. This method can be seen as black-box testing and the goal is to verify that UMA-RR protocol specifications and this testing implementation are not in conflict. This can be done by initiating e.g. a discovery procedure and verifying that the correct messages

and parameters are used. Also state transitions and timer operations are verified so that they comply with the functionality defined in UMA protocol specifications. Because of the nature of testing environment, other protocol testing methods, like interoperability test or performance test, are not carried out at this time. Next chapter gives a description of the used testing environment.

4.1 CVOPS

CVOPS stands for C Virtual OPerating System and it used for implementing the communication protocols and tester software. It supports both C and C++ languages and its goals are portability of code and performance. CVOPS consist of function and data type libraries, run-time environment and implementation framework.

Libraries define the basic means for creating the implementation of the tested protocol. Main features of libraries include the following:

- Timer services
- Frame services for representing and handling of octet strings
- Message sending and receiving
- Protocol addresses
- Concurrent connections

CVOPS framework has built-in state machine, which is called Extended Finite State Automation, EFSA. This state machine supports defined states, transitions, message handling, timers and functions for different operations. EFSA uses these C- or C++ -functions for various purposes, e.g. service logic operations and encoding or decoding of messages.

4.1.1 Virtual tasks

CVOPS uses virtual tasks for protocol implementation. These tasks can be referred as objects consisting of data and functions. Virtual tasks communicate by sending messages

to each other. CVOPS has services for message handling, e.g. for encoding and decoding of messages, message queues and parameter handling. The Figure 18 shows en example of CVOPS based protocol implementation with three virtual tasks. In the figure, virtual task A sends a message to virtual task B by using a CVOPS service, e.g. sendMessage(). SendMessage() stores the message to temporary storage queue and later at appropriate time takes to message from queue, and executes a special function of the receiver virtual task to handle the message.

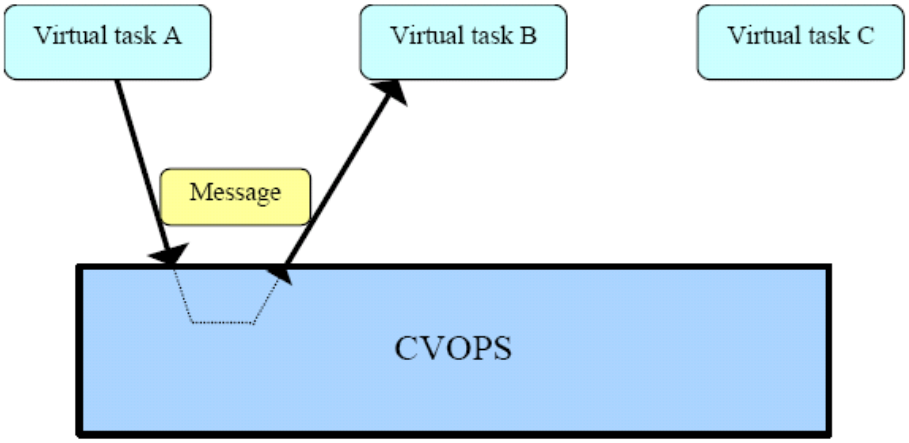

Figure 18. Virtual tasks and message sending

CVOPS isolates the protocol implementations (virtual tasks) from the operating system. This applies so that virtual tasks only use the service of CVOPS and CVOPS uses operating system services, when necessary. For example, protocol implementations should use CVOPS services for starting timers and not directly the timer services provided by the operating system. The Figure 19 shows the concept of virtual tasks and CVOPS.

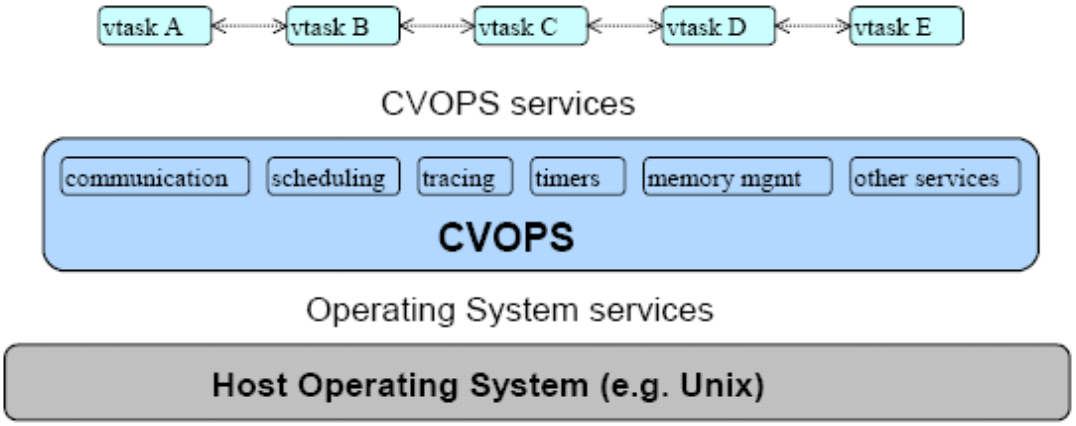

Figure 19. Concept of virtual tasks and CVOPS

Virtual tasks are type of VTASK, which is a C language struct. This struct has a pointer to variable structure, which holds the internal variables of a VTASK. Also the current state in state automation is stored in the struct. Variable *type* is a pointer used to point to VTASK type specific structure. This structure holds information shared by VTASKs of the same type, e.g. all the virtual tasks that implement network- or link-layer.

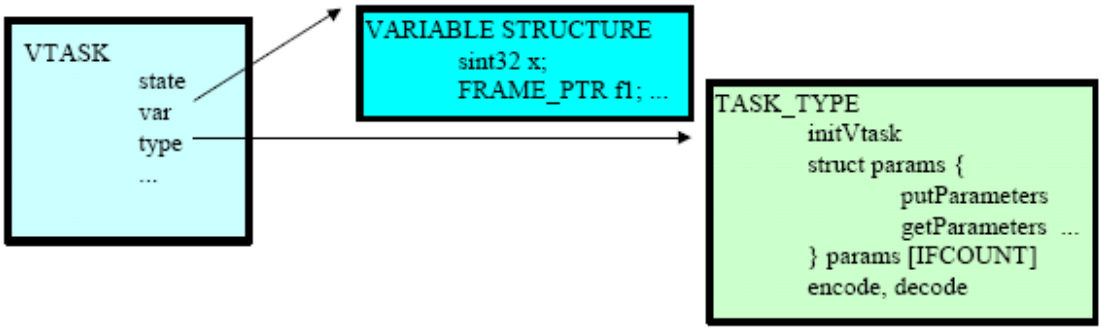

Figure 20. Structure of virtual task

Virtual task can have one or several interfaces. Communication between virtual tasks can only occur through an interface, so any messages sent or received by a virtual task must have been specified in interface definitions. The protocol implementation cannot use own names for interfaces, but must use the predefined names, e.g. “UP”,”DOWN”,”RES1” etc. Chapter 4.1.3 gives more information about the interface file and also more information on other file types.

4.1.2 Primitives, PDUs and frames

Primitives and PDUs are used to send information to other entities. Primitives are data units that are sent to upper or lower layer entities. PDUs are internal constants that are used in information passing to peer entities. The Figure 21 shows the concept of service primitives and PDUs.

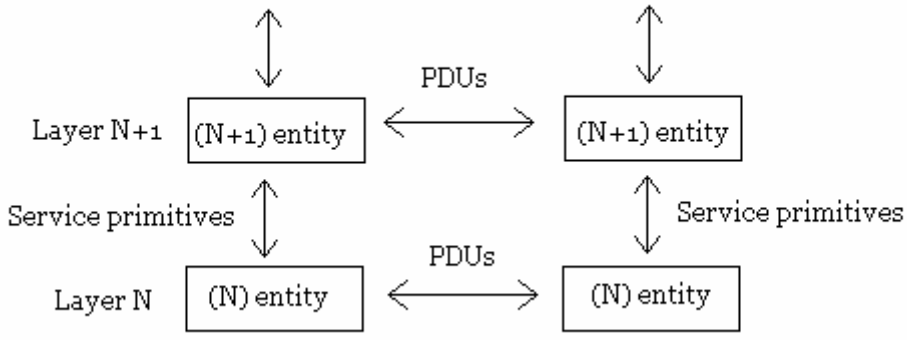

Figure 21. Service primitives and PDUs

Frames are used in CVOPS for storing octet strings, e.g. PDUs, in an efficient way. Internally frames are implemented as lists of data block. CVOPS offers many functions for frame handling, like inserting and reading bytes from frame, splitting a frame into two frames or combining two frames into one frame. Frames can be accessed from EFSA or from C-functions. C-functions also define the needed decoding and encoding functions for frames.

4.1.3 File identifiers

For every protocol implementation, there are several defined file types that must be implemented to create a fully operational CVOPS implementation. These files define the basic operations, like state machine automation or interfaces to other virtual tasks. Next is the list of typical file types that are used in CVOPS implementations.

4.3.1.1 Layer interfaces: ”*_if.h”

Interface file describes the service interface between two layers residing in different levels. Basically this concerns the service primitives that two connected layers use in communication between them. Interface files also define the parameter block, which is typically used by all the service primitives in the same interface. The following figure shows how the interface files are used in implementation. The example in Figure 22 consists of transport layer, network layer and link layer. Between each protocol layer is an interface file that defines the used primitives and parameter block used in communication between layers.

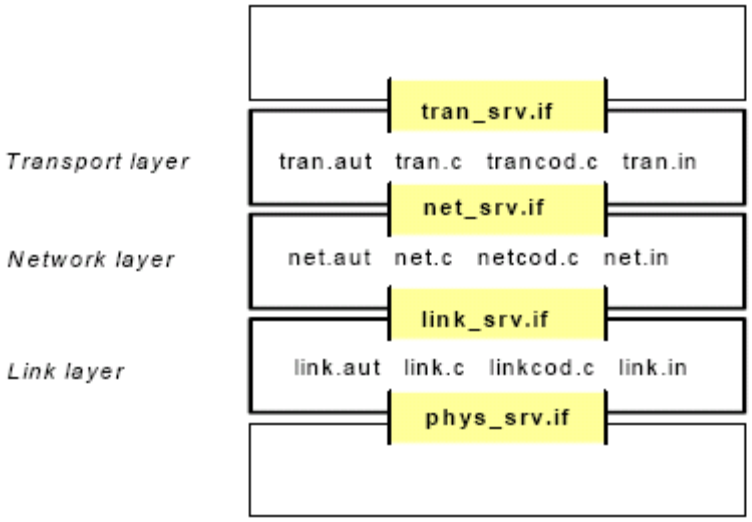

Figure 22. Usage of interface definitions

4.1.3.2 Layer internals: ”*_in.h”

Layer internal files define the internal structure of a layer. These files include the PDUs used in peer layer communication, states used in EFSA, internal variables and symbolic constants of a layer and also timers.

4.1.3.3 State machine: ”*.aut”

The state machine consists of state-input-actions triplets, which are also called state transitions. At any time, the state machine is at one state. At this state, it accept some

defined inputs, does various tasks (e.g. copying parameters to correct blocks, setting timers or decoding frames) and may send outputs or execute transitions to other states. The state machine file describes the actual behaviour of a layer entity (protocol).

4.1.3.4 Other files types

A CVOPS implementation is not limited to consist of only previously described file types. Usually there are at least some files made with C- or C++ -language that consist of different functions used in CVOPS implementations, e.g. encoding and decoding functions of PDUs and frames. Also virtual tasks and connections between VTasks are usually defined in an own association file.

4.2 UMA-RR CVOPS testing implementation

This chapter deals with protocol testing implementation that is used in this thesis work. The implementation consists of two layers, the transport layer and the network layer. The transport layer implements the functionality of the tested UMA-RR protocol. It uses the network layer to relay messages between endpoints. As this testing implementation concentrates only on the testing of UMA-RR, the network layer does not fully implement the actual lower layer protocol (TCP). For the purpose of this thesis work, lower layer relays message of UMA-RR protocol in a simplest possible way.

Both layers are separate virtual tasks with different functions, variables and state machines. The following Figure 23 shows the used layers and interfaces between them. The urr_if.h is used to define the primitives between the user layer and UMA-RR layer. The netw_if.h is used to define the primitives used between UMA-RR layer and network layer. Both interface definitions also contain parameter block variable list. Parameter block is a structure, which contains the needed parameters in different primitives of an interface.

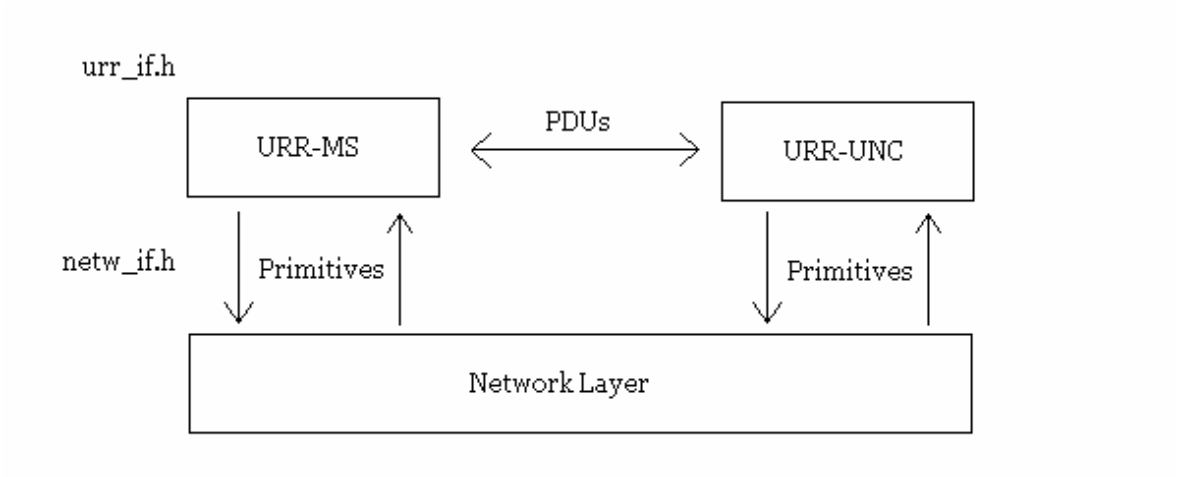

Figure 23. Layer and Interfaces

As one goal of the testing implementation is portability, the transport layer protocol, implementing the UMA-RR, is made to match the specifications very precisely. As mentioned in previous chapter of protocol testing, the conformance of a protocol is verified against the specifications. Any deviation between the implementation and specifications can be seen as an error, which can result to unwanted behaviour or complete failure of a system. The purpose is not, however, to fully implement the whole protocol, as it would require very much time and effort. This protocol implementation at this stage supports UMA discovery procedure and UMA registration procedure.

Later implementations could add complete versions of lower layer protocols, like e.g. TCP, IP and IPSec protocols for other testing purposes. This could be done by adding the virtual tasks of other needed protocols and changing the interface definitions of UMA-RR testing implementation. The internal behaviour of UMA-RR layer doesn't need any changes because of the layered structure of the protocol stack.

4.2.1 States and timers

Each Vtask is at all times in one defined state. At this defined state, the Vtask can accept certain inputs and make some operations according to service logic defined in state automation.

The test implementation of UMA-RR begins with identifying the basic states of the protocol. UMA-RR has for states that are described in [UMAp05]. The Figure 24 shows the states and transition.

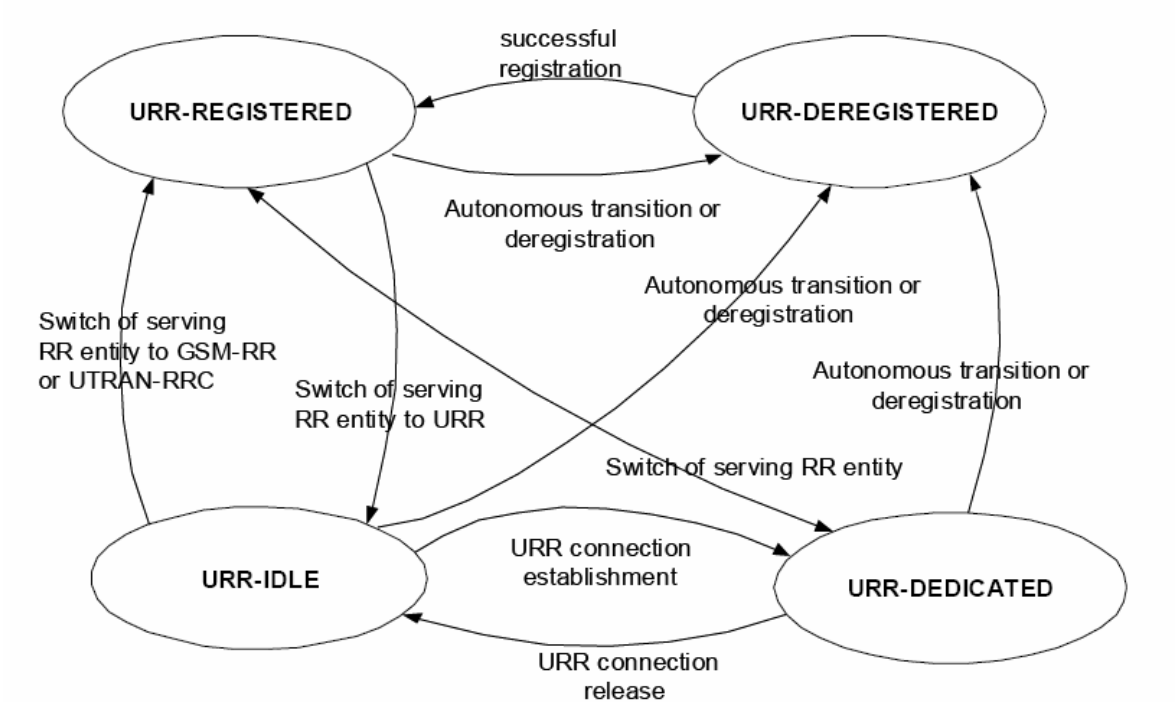

Figure 24. UMA-RR protocol states and transitions

Four listed states are created into umarr_in.h file. This file also contains used PDUs, timers and variable structure. Needed timers are listed in specifications and they the following:

UMA Discovery procedure:	TU3901	initial value: 30 seconds
	TU3902	initial value: random
	TU3903	initial value: 60 seconds

UMA Registration procedure:	TU3904	initial value: 30 seconds
	TU3905	initial value: 10 seconds
	TU3907	initial value: random

The values of timers TU3902 and TU3907 depend on the random value that the MS creates after receiving reject message from the UNC signifying network congestion. This timer is used with the resending of messages after certain time period to avoid congestion that may be caused by multiple resends at the same time interval.

4.2.2 Messages and message sequences

Testing program implements two procedures described in specifications [UMAp05]. These procedures are UMA Discovery Procedure and UMA Registration Procedure. Following Figures 25 and 26 show the signalling sequence of those two procedures.

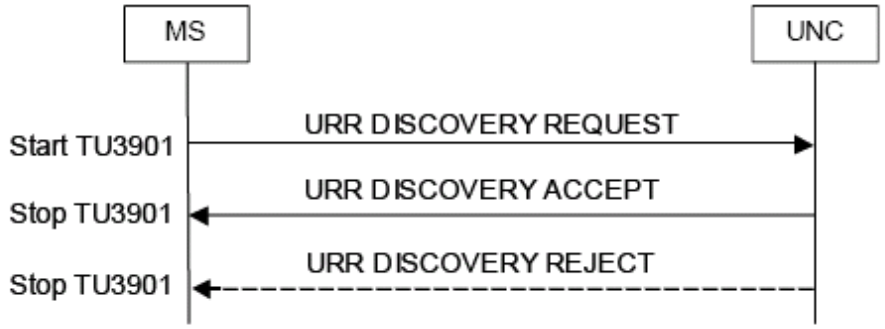

Figure 25. URR Discovery procedure [Uma05p]

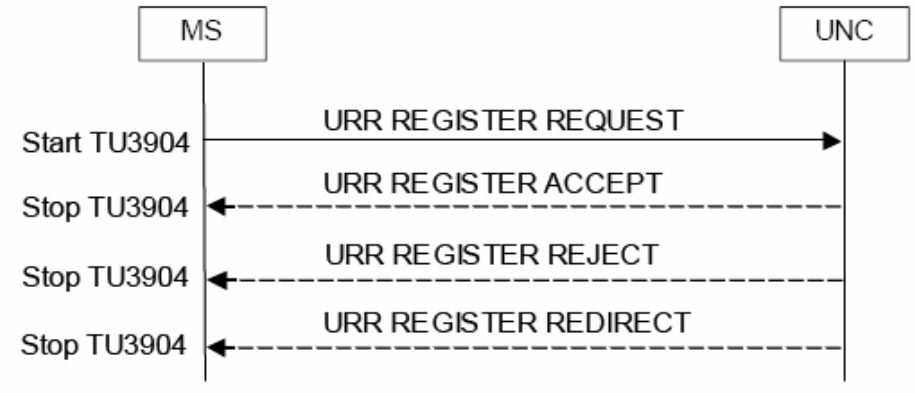

Figure 26. URR Registration Procedure [Uma05p]

Implementation uses the same signalling sequence with messages that have the same internal structure as defined in specifications. The complete list the messages and their structure can be found at UMA protocol specifications [Uma05p]. Messages are encoded

using TLV encoding. TLV encoding means Type, Length and Value encoding, where information elements are constructed so that they always consist of information element identifier, length identifier and actual value part. Example of TLV constructed information element can be as presented in Figure 27.

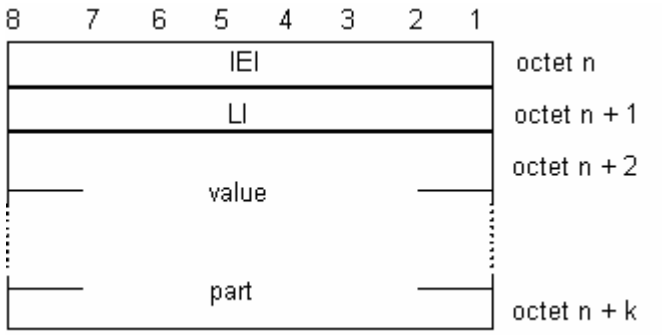

Figure 27. The example of TLV constructed IE. [TS24.007]

In UMA, the TLV encoded messages are a bit different. Each message has a header and IEIs. The structure of URR message header is described in Table 4.

Table 4. URR message header.

IEI	Information Element	Type/Reference	Presence	Format	Length	Value	Notes
	Length Indicator	**Length Indicator 11.1.1.1**	M	V	2		
	UMA RR_C Protocol Discriminator	**Protocol Discriminator 11.1.1.2**	M	V	½	0000	
	Skip Indicator	**Skip Indicator 11.1.1.3**	M	V	½	0000	
	URR DISCOVERY REQUEST Message Type	**Message Type 11.1.1.4**	M	V	1		

Length Indicator has size of 2 octets and it tells the length of the message excluding the 2 octets from itself. URR Protocol Discriminator is ½ octets long and it specifies the upper layer protocol to which the message belongs. Skip Indicator is also ½ octets long and it simply tells if received message should be ignored. Any message with Skip Indicator deviating from value 0 shall be ignored. Last header field tells the URR message type.

Every URR message, like Discovery Request or Registration Accept has its own value specified in this field. The testing implementation uses these values to identify which message is carried inside a frame.

IEIs after the header specify the actual content (parameters) of the message. Each IEI has its own identifier (value of byte), length indicator and information fields. The specifications [Uma05p] give the complete list of every IEIs with every URR message and their contents. IEI can be seen as a struct data type in any programming language. Also each message is a struct, so IEIs can be seen as structs inside of a struct.

This protocol implementation uses the exact same type of messages that are listed in specifications. Parameters are listed in variable structure of each Vtask. The construction of message is done by using a frame data type, which stores the octet strings of a message. Each message has its own encode and decode functions, which create the message by either using user-given or default parameter values. Parameter values can be read from a macrofile instead of calling the function askURRServiceParameters() in Vtask. Specifications define which parameters are mandatory, conditional or optional. Optional parameters are not used with this testing implementation. Conditional parameters are used as defined in specifications. Usually it depends on the test case which parameters must be included in a message.

For the simplicity of the testing program, not all parameters are adjustable by the user. The two implemented test cases have parameters, which values doesn't change in testing scenarios, and they are therefore initialized into default values. Later modifications for testing implementation will add the feature for modifying these values. It would only require some modifications to askURRServiceParameter() - function and to the macrofile used.

4.2.3 Encoding and decoding functions

When a message is sent to other Vtask, it is encoded. Encoding of a message is done by calling a appropriate encoding function which inserts the message header and IEIs into a

frame. Message encoding function is a main encoding function, which checks the type of message (PDU). After checking the type, each IEI has its own encoding function. Table 5 shows an example of an IEI used to encode IP address. The structure of IP address IEI is presented as defined in UMA specifications.

Table 5. The structure of IP address [Uma05p]

8	7	6	5	4	3	2	1	
	IP address IEI							octet 1
Length of IP address IE contents								octet 2
IP Address type								octet 3
Address information MSB								octet 4
								...
Address information LSB								octet n

Next is an example taken from the code, which shows the function implementation and the function call.

Function call:

```
encIP (frame, &var->sysparms->sgw_addr, UMA_IEI_SGW_IP);
```

Function implementation:

```
void encIP (FRAME_PTR *frame, struct sockaddr_in *sa, BYTE_TYPE iei)
{
  UINT32 ip = sa->sin_addr.s_addr;

/* This breaks UINT32 datatype into 4 eight bit datatypes (byte) and
   stores bytes into the frame */
  putFirstByte (frame, ip>>24); /* IP address LSB */
  putFirstByte (frame, ip>>16);
  putFirstByte (frame, ip>>8);
  putFirstByte (frame, ip);     /* IP address MSB */

  /* IP Address type == version 4, hardcoded */
  putFirstByte (frame, 0x21);

  /* Length of IP address IEI contents, length is global variable */
  length = 5;
  putFirstByte (frame, length);

  /* IP address IEI */
  putFirstByte (frame, iei);

}
```

As the upper example shows, the encoding function takes for inputs the following parameters:

- FRAME_PTR *frame, which is pointer to frame data type that stores the octets
- struct sockaddr_in *sa, which is a pointer to variable string parameters of Vtask.
- BYTE_TYPE iei, which is an enumerated identifier for IEI.

As frames are used to transport protocol message information, the receiving side must be able to interpret the incoming messages and its parameters correctly. This is done with the usage of decode-functions. Decode functions work as opposite of encoding functions. They check the incoming frame, each byte at the time, and extract the message parameters stored inside the frame. These parameters are stored into Vtask local variable structure and verified that they conform with protocol specifications.

The other important aspect is to verify the internal behaviour of the testing implementation so that state transitions, and also frame and parameter handling, work as intended. This verification can be seen as white-box testing of the protocol implementation as the behaviour of the internal structure is being tested. The verification is made with debugging and tracing of parameter values in certain defined situations. Implementation has three debug-variables, which are used to print various parameter values, function calls and other common information about the behaviour of the implementation. Also the state transitions and timers are checked and verified. These can be verified from the printouts of EFSA. The example of IP address decoding function and function call are presented next.

Function call:

```
decIP (frame, &var->sgw_addr);
```

Function implementation:

```
int decIP (FRAME_PTR *frame, struct sockaddr_in *sa)
{
  BYTE_TYPE iei;
  /* Get first byte from frame */
  iei = getFirstByte (frame);
```

```
  /* Check which IEI is inside frame */
  switch (iei) {
  case UMA_IEI_SGW_IP:
    if (debug1) traceLine ("UNC SGW IP:\n");
    break;
  case UMA_IEI_UNC_IP:
    if (debug1) traceLine ("UNC IP:\n");
    break;
  default:
    traceLine ("decIP(): invalid IEI!\n");
    return 0;
  }

  /* Get the length of IEI */
  length = getFirstByte (frame);
  if (length != 5) {
    traceLine ("decIP(): invalid length, only v4 supported\n");
    return 0;
  }

  /* Check IP protocol version */
  if (getFirstByte (frame) != 0x21) {
    traceLine ("decIP():  only v4 supported\n");
    return 0;
  } else {
    sa->sin_family = AF_INET;
  }
  /* Copy IP address to struct sockaddr_in, one byte at the time */
  sa->sin_addr.s_addr = (getFirstByte (frame) << 24) & 0xff000000;
  sa->sin_addr.s_addr |= (getFirstByte (frame) << 16) & 0x00ff0000;
  sa->sin_addr.s_addr |= (getFirstByte (frame) << 8)  & 0x0000ff00;
  sa->sin_addr.s_addr |=   getFirstByte (frame);

  if (debug2){
   tmp = inet_ntoa(sa->sin_addr.s_addr);
   printf("IPaddress: %s\n",tmp);
  }
  return 1;
}
```

This decoding function takes in two parameters, which are pointers to incoming frame and pointer to Vtask variable string that holds the IP address. Information is extracted from the frame and put into socket address struct of variable string. Other decoding functions operate in a similar way. Function parameters of decoding functions contain always a framepointer and, depending on the decoded message, pointers to variable string elements that are used to store the correct IEIs from a frame.

4.3 Test case execution

CVOPS platform is run on a remote workstation with operating system of Red Hat Linux version 3.2.2-5. CVOPS user interface is used from a PC workstation, including Windows 2000 as operating system, with a remote telnet connection.

CVOPS testing implementation is compiled with a makefile, which has the necessary associations for needed files and also it specifies the used compiler and file paths. After compile, the testing implementation can be started by typing the command `./umarr`. At first, the CVOPS reads initialization parameters from a macrofile. This sets the correct paths for source codes, e.g. for symbol tables and state automation. After that, the test cases can be run.

As the test implementation doesn't include any upper layers, the service parameters from those layers are stored directly to the internal variables. Before CVOPS gets the parameters from incoming message with function getServiceParameters(), the function askServiceParameters() can be called, which asks the defined parameters from the user instead and stores them to virtual tasks internal variables. The askServiceParameters() - function can also read the parameters from a macrofile and it is used with this testing implementation, as it is a faster way to set the correct parameters.

User-given parameters include the IP- address or FQDN - address of the default UNC and default SGW. Other parameters are IMSI, Cell-ID, MCC and MNC, which normally would be included from the MS or BTS. These parameters affect on procedures of discovery and registration.

Test cases are defined in a macrofile. For each test case, the macrofile contains following definitions:

- Service primitive invoking from correct VTask
- Log file handling
- Setting of user-defined parameters

Each URR message has its own service primitive defined in the interface file. These primitives can be used in state automation and they can be seen as messages from upper layers. The user invokes service primitives according to testcases executed.

4.3.1 Conformance verification

The Conformance verification methods of the UMA-RR protocol testing implementation are described in the conformance-testing plan, which holds the information of used test cases. As it is not necessary to inspect every single test case here, an example is presented at the appendix A. Test cases are based on the UMA Conformance Test Specification 1.0.4 created by the UMA consortium [UMAt05] with some modifications, as the testing environment is not suitable for complete conformance resolution testing.

The basic concept of conformance testing is presented in Figure 28. The implementation under test is tested with different test sequences to verify that implementation is constructed to conform with the reference specifications. Test cases are executed according to test plan and results are compared to the acceptance requirements and protocol specifications.

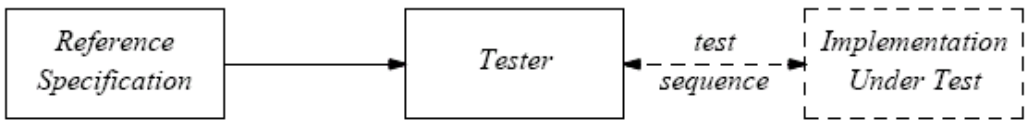

Figure 28. The conformance verification concept

The complete list of executed test cases set to discovery and registration procedures are presented in appendix B. These cases contain both successful and unsuccessful discovery procedures. The test plan contains information of requirements and acceptable results for test cases. Requirements for the test cases conformance verification include the following:

- Correct message sequence

- Correct message parameters
- Correct timer handling and state transitions

Also for every test case, the repeated test execution must give the same results, when exexuted with same start-up state and conditions.

4.4 Analysis of results and implementation

The results of executed test cases are also listed in appendix B. The test case executions show that the results passed the requirements set in testing plan. So in conclusion, the protocol implementation conformed with the specifications of UMA according to test case executions of discovery and registration procedures.

Each test case was executed few times to verify that implementation doesn't have any substantial memory allocation problems or any other mishandling of parameters. After the execution of a test case, the log file was analyzed to see if correct messages were used and the overall behaviour of the protocol implementation complies with the protocol specifications.

An example of log-file printout can be seen in appendix C. The log file shows quite comprehensively the message sequence and parameters that are used in a test case. When comparing the results obtained from the log-file to the protocol specifications, the operational behaviour of the testing implementation can be verified.

The implementation process itself was a rather meticulous activity balancing between the planning, coding and verifying of functionality from the protocol specifications. First versions of testing implementation had some conflicts with the specifications, but they were gradually analyzed and fixed. Testing of the latest version of protocol implementation didn't reveal any noticeable bugs or errors and test cases could be executed as planned.

In future the UMA-RR testing implementation can be used with other systems that need it for different testing purposes. The encoding and decoding functions of URR messages use

the CVOPS data type of frame, which is simply an octet string of variable length. Any other system that supports this data type can use the encoding and/or decoding functions of URR messages. The portability of testing implementation is fairly easy in CVOPS protocol implementations. The interface definitions can be modified to support any other upper layer protocol with the usage of propriate service primitives. Also lower layer protocols can be modified as the control information of URR protocol is encoded into a frame, which travels transparently between lower layer protocols. The layered structure model used in various communication protocols is a good asset when portability is considered.

CVOPS in general is quite efficient tool for creating communication protocol implementations. Especially for testing purposes, the benefits come directly from automated state machine, handling of timers and automated sending and receiving of messages. Also the message construction and parameter handling with CVOPS is more easier when compared to implementations that are created without a proper protocol implementation framework.

5. CONCLUSIONS

When reading the specifications, the Unlicensed Mobile Access technology seems to be ready and well-thought new access method for mobile subscribers. Unlike other fixed-mobile convergence solutions, the UMA technology is tightly linked to existing mobile core network. Regardless of the air interface method used, the mobile stations end up using the same 2G or 3G core network. As the UMA is not a SIP-base VoIP technology, the core network is still old-fashioned circuit-switched GSM network. This is a positive driving force when costs of deploying UMA are considered. On the other hand, the analysts have said that the most challenging component of UMA based solutions is the handset, because of cost, size and battery-life limitations.

One important step for solidifying the UMA concept among different technologies has been seen, as the 3GPP has approved UMA as a preliminary standard. The participating companies of UMA consortium have contributed the UMA specifications to 3GPP as part of the 3GPP working item called the “Generic Access to A/Gb Interfaces”. As a result, the companies in UMA Consortium have since then been working with the 3GPP for continuing of developing the UMA standard further.

As for the aspects of testing, this thesis work has shown, that protocol and software testing in general are rather important phases when dealing with the creation of mobile protocol systems or other communication software. As the systems and devices have grown to be quite complex, there are simply just so many aspects that need to be taken into account, when software and especially protocols are implemented. One little flick with a single bit can cause the whole system to work improperly. Without carefully thought and constructed testing plan and methods the software quality could decrease quite dramatically. Also any non-working software will increase the overall cost of development as in terms of error-correction and maintenance. The consumers will quite surely neglect any mobile device with software errors and bugs, as the end-users of these devices have accustomed to high quality.

The construction of testing implementation can be a very time-consuming procedure. The conformance requirements must be reviewed with extreme carefulness as the deviation with the testing software and specifications can result in false results or completely incorrect behaviour of the testing implementation or the whole testing environment.

SOURCES

[And01] GPRS and 3G Wireless Applications, Christoffer Andersson, John Wiley & Sons Inc., 2001, ISBN 0-471-41405-0.

[Ala03] Teemu Alakoski, Avainten vaihto ja jakelu IPSec-järjestelmässä, Masters Thesis, Tampere University of Technology, 2003.

[Beiz90] Software Testing Techniques 2nd edition, Boris Beizer, International Thomson Computer Press, ISBN 1-850-328-803, 1990.

[Craig03] Rick D. Craig and Stefan P. Jaskiel, Systematic Software Testing, Artech House Publishers, 2002, ISBN 1-58053-508-9.

[Gollm00] Computer Security, Dieter Gollmann, John Wiley & Sons Inc., 2000, ISBN 0-471-97844-2.

[Hai01] Ilkka Haikala & Jukka Märijärvi, Ohjelmistotuotanto 7. painos, Talentum Media Oy, 2001, ISBN 951-762-769-6.

[Hav03a] Haverinen H. & Salowey J., Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM), Internet Engineering Task Force draft-haverinen-pppext-eap-sim-16.txt, [http://ietfreport.isoc.org/ids/draft-haverinen-pppext-eap-sim-16.txt], referred on 11.06.2005.

[Hav03b] Haverinen H. & Arkko J., Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA), Internet Engineering Task Force draft-arkko-pppext-eap-aka-15.txt, [http://ietfreport.isoc.org/ids/draft-arkko-pppext-eap-aka-15.txt], referred on 11.06.2005.

[ITUG.732] ITU-T G.732, Characteristics of Primary PCM Multiplex Equipment Operating at 2048 kbit/s, ITU-T Recommendation G.732, 1993.

[Mouly92] Mouly M. & Pautet M-B., The GSM System for Mobile Communications, ISBN 2-9507190-0-7, 1992.

[Mus00] Jarmo Mustonen, Puhelinverkon ja Internetin yhdyskäytävän tietoturvallisuushaavoittuvaisuus, Masters Thesis, Tampere University of Technology, 2000.

[Ora00] Jyrki Oraskari, Bluetooth versus WLAN IEEE 802.11x, Internet Document, [http://users.tkk.fi/~joraskur/BT2.pdf], referred on 27.07.2005.

[RFC2402] Kent S. & Atkinson R., IP Authentication Header, RFC2402, 1998.

[RFC2406] Kent S. & Atkinson R., IP Encapsulating Security Payload (ESP), RFC2406, 1998.

[Royer93] Thomas C. Royer, Software Testing Management - Life on the Critical Path, Prentice Hall Inc., 1993, ISBN 0-13-532987-6.

[S0055-A] Association of Radio Industries and Businesses, STD-T64-S.S0055-A v2.0, Enchaned Cryptographic Algorithms, 2005.

[Sig01a] Specification of the Bluetooth System, v1.1, Volume 1: Core. Bluetooth Special Interest Group, 2001.

[Sig01b] Specification of the Bluetooth System, v1.1, Volume 2: Profiles. Bluetooth Special Interest Group, 2001.

[Sig01c] Bluetooth Network Encapsulation Protocol (BNEP) Specification version 1.0, Bluetooth Special Interest Group, 2001.

[Sig04] Specification of the Bluetooth System, v2.0, Bluetooth specification 2.0 + EDR, Bluetooth Special Interest Group, 2004.

[Tak00] Kimmo Takanen, An Estimation Method of the Number of Recommended Test Cases for Protocol Conformance Testing, Licenciate Thesis, University of Oulu, 2000.

[Tan03] Computer Networks 4th edition, Andrew S. Tanenbaum, Pearson Education Inc., 2003, ISBN 0-13-038488-7.

[TS08.71] 3GPP TS 08.71 V8.5-0, Location Service (LCS);Serving Mobile Location Centre - Base Station System (SMLC-BSS) interface, Layer 3 specification (Release 1999), 3rd Generation Partner Project, 2002.

[TS09.31] 3GPP TS 09.31 V8.7-0, Location Services (LCS);Base Station System Application Part LCS Extension (BSSAP-LE) (Release 1999), 3rd Generation Partnership Project, 2004.

[TS24.007] 3GPP TS 24.007 V4.4.0, Mobile radio interface signalling layer 3, general aspects, 3rd Generation Partnership Project, 2005.

[TS33.102] 3GPP TS 33.102 V5.6.0, 3G Security; Security Architecture, 3rd Generation Partnership Project, 2005.

[TS43.020] 3GPP TS 43.020 V6.1.0, Security Related Network Functions, 3rd Generation Partnership Project, 2005.

[TS101.369] ETSI TS 101.369 V6.3-0, Digital cellular telecommunications system (Phase 2+) (GSM); Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol, European Telecommunications Standards Institute, 1999.

[Std829] IEEE std829-1998, IEEE stardard for Software Test Documentation, Institute of Electrical and Electronic Engineers, 1998.

[Sys00] Systra GSM System Training, Nokia Networks oy, 2000.

[UMAa05] Unlicensed Mobile Access Architecture, UMA Consortium 2005, [http://www.umatechnology.org/specifications/index.htm], referred on 18.05.2005 and later.

[UMAp05] Unlicensed Mobile Access Protocols, UMA Consortium 2005, [http://www.umatechnology.org/specifications/index.htm], referred on 25.05.2005 and later.

[UMAt05] Unlicensed Mobile Access Mobile Conformance Test Specifications 1.0.4, UMA Consortium 2005, [http://www.umatechnology.org/specifications/index.htm], referred on 15.10.2005 and later.

[UMAu05] Unlicensed Mobile Access User Perspective, UMA Consortium 2005, [http://www.umatechnology.org/specifications/index.htm], referred on 22.05.2005 and later.

[Wal02] Bernhard H. Walke, Mobile Radio Network; Networking, Protocols and Traffic Performance 2nd edition, John Wiley & Sons, ISBN 0-47-149902-1.

APPENDIX A

An example of discovery procedure test case

Name: CONF_URR05_1a

Events:

1. Edit the test macro to contain the following requirements:
   a. Log file opening for test case conf_urr05_1a.
   b. IMSI for MS.
   c. IP-addresses for UNC and SGW.
   d. Parameter indicating GERAN coverage is set to on.
   e. MCC and MNC for MS used for discovery acceptance at the UNC side.
   f. Debug1, debug2 and debug3 are set to value 1.
   g. Parameters communication port, release indicator, cell-id are adjusted with random value within the parameters value ranges.
2. Start URR protocol tester.
3. Set trace for all events with command: set trace all all.
4. Run the test macro for conf_urr05_1a.
5. Close the log file.

Expected results:

1. The correct message sequence for successful case of URR Discovery.
   a. MS side sends UMA DISCOVERY REQUEST (id 0x01) to UNC.
   b. UNC replies with UMA DISCOVERY ACCEPTED (id 0x02) to MS.
2. Discovery is accepted at the UNC side with correct IMSI value.
   a. MCC and MNC are correct (IMSI allowed).

b. No network congestion.

3. Internal parameters are correct on both virtual tasks.

4. Encoded parameters have the same value after decode.

5. No memory exceptions or other errors.

6. State transitions correspond the specifications.

7. Timer settings correspond the specifications.

8. Concurrent test case executions don’t change the results.

APPENDIX B

Test cases and results

TEST ID	**CONF_URR05_1a**
DESCRIPTION	"Discovery accepted, MS holds provisioning UNC IP, MS holds provisioning SGW IP, GSM coverage is on, rove-out from GSM cell."
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_1b**
DESCRIPTION	"Discovery accepted, MS holds provisioning UNC IP, MS holds provisioning SGW IP, UMTS coverage is on, rove-out from UMTS cell."
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. UTRAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_2a**
DESCRIPTION	"Discovery accepted, MS holds provisioning UNC FQDN, MS holds provisioning SGW FQDN, GSM coverage is on, rove-out from GSM cell."
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_2b**
DESCRIPTION	"Discovery accepted, MS holds provisioning UNC FQDN, MS holds provisioning SGW DQFN, UMTS coverage is on, rove-out from UMTS cell."
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. UTRAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_3a**
DESCRIPTION	“Discovery accepted, MS does not hold information of provisioning UNC or SGW, GSM coverage is on, rove-out from GSM cell.”
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. Derive FQDN from the IMSI, GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_4a**
DESCRIPTION	“Discovery rejected, network congestion, GSM coverage is on, rove-out from GSM cell, UNC replies with discovery rejected message, indicating network congestion. MS resends discovery request after timer TU3902 expires.”
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. GERAN coverage on, congestion parameter set on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_4b**
DESCRIPTION	“Discovery rejected, IMSI not allowed, GSM coverage is on, rove-out from GSM cell, UNC replies with discovery rejected message, indicating that IMSI is not allowed. MS stops discovery procedure.”
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. MNC or MCC set to mismatch with IMSI, GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_5a**
DESCRIPTION	“Registration accepted, Serving UNC known, GSM coverage is on, rove-out from GSM cell.”
PRECONDITION	MS in state Discovered, not registered. Internal parameters set to correct values. GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	CONF_URR05_5b
DESCRIPTION	“Registration accepted, Serving UNC known, UMTS coverage is on, rove-out from UTRAN cell.”
PRECONDITION	MS in state Discovered, not registered. Internal parameters set to correct values. UTRAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	CONF_URR05_5c
DESCRIPTION	“Registration accepted, Serving UNC known, MS is not in GSM or UMTS coverage.”
PRECONDITION	MS in state Discovered, not registered. Internal parameters set to correct values.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	CONF_URR05_6a
DESCRIPTION	“Registration accepted, MS holds provisioning UNC IP, MS holds provisioning SGW FQDN, GSM coverage is on, rove-out from GERAN cell.”
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	CONF_URR05_6b
DESCRIPTION	“Registration accepted, MS holds provisioning UNC FQDN, MS holds provisioning SGW IP, GSM coverage is on, rove-out from GERAN cell.”
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. GERAN coverage on.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	CONF_URR05_7a
DESCRIPTION	“Registration rejected, network congestion. MS tries to register, rove-out from GERAN cell, UNC replies with Register reject indicating network congestion”

PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. Network congestion is set on at UNC.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

TEST ID	**CONF_URR05_7b**
DESCRIPTION	“Registration rejected, AP not allowed. MS tries to register, rove-out from GERAN cell, UNC replies with Register reject indicating Access point is not allowed”
PRECONDITION	MS in state IDLE, not registered. Internal parameters set to correct values. AP block is set on at UNC.
RESULT	OK
TYPE	[CVOPS] Conformance, URR.

APPENDIX C

Log-file of successful Discovery Procedure (test case CONF_URR05_1a):

```
/************************************************************
 * This file was generated by CVOPS release 7.3.0.
 * Generation time:               2006 Jan 6, 16:39:18
 ************************************************************/


: run

: set trans_2 var debug1
new value:1

: set trans_2 var debug2
new value:1

: set trans_2 var debug3
new value:1

: set trans_2 var paramMCC
new value:843

: set trans_2 var paramMNC
new value:74

: trans_1 u_disc_req

 IP address(1) or FQDN address(2):1

 Insert UNC IP (xxx.xxx.xxx.xxx):123.45.125.35

Insert UNC com port:1060

 Insert SGW IP (xxx.xxx.xxx.xxx):175.115.241.76

Insert UNC SGW com port:2111

 Give debug1 value: 1

 Give debug2 value: 1

 Give debug3 value: 1

 Give discovery rejected-value: 1

 Give release_indicator: 0

 Give IMSI for MS: 84374672348

 Geran(1) / Utran(2) -coverage? (0=no coverage)  1

 Give Cell ID : 843

 Give MCC (when GSM coverage is on): 74
```

```
 Give MNC (when GSM coverage is on): 68

 Give LAC (when GSM coverage is on): 4

 Give UNC SYSTEM MCC: 55

 Give UNC SYSTEM MNC: 65

 Network congestion : 0

 IMSI not allowed? : 0



DEBUG tranEncode
 deb1=1, deb2=1, deb3=1
  Alternative commands:
   commandMode, run(D), ?, .

: trans_1 u_disc_req (default = run(D))


*************************************************************************

 MESSAGE

 source     : USER
 destination: trans_1
 primitive  : u_disc_req     to receiver's UP interface
 src cepid  : -
 dest cepid : -
 time       : 6.1.2006 16.39:18.821.114
 parameters :


 VTASK BEFORE RECEIVING THE MESSAGE:
 vtask      : trans_1
 state      : idle
 variables  : congestion       :        0 debug1               :        0
              debug2           :        0 debug3               :        0
              disc_rejected    :        0 geran_cov            :        0
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :        0
              paramMNC         :        0 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 timers     :

 VTASK AFTER GETPARAMETERS :
 vtask      : trans_1
 state      : idle
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        1 geran_cov            :        1
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :        0
              paramMNC         :        0 prot_disc_and_skip_i :        0
```

```
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 state      : idle
 input      : UP.u_disc_req
 actions    :

                { PEER.DISC_REQ;
      DEBUG: reg_rej:1
            EncIP UNC
      DEBUG: encIP
            EncIP SGW
      DEBUG: encIP
      DEBUG: encRedirCounter
      DEBUG: encRegRejCause
      DEBUG: encRAC
      DEBUG: encLAI
            MCC = 74, MNC = 68, LAC = 4
      DEBUG: encUMACellID
      DEBUG: encGSMcoverageInd
      DEBUG: encReleaseInd
      DEBUG: encUMAMobIdt
IMSILENGTH: 11
1. Element of IMSI: 132

2. Element of IMSI: 50

3. Element of IMSI: 118

4. Element of IMSI: 71

5. Element of IMSI: 52

6. Frame of IMSI: 137
DOWN.t_dtr;
#DEBUG# beginning at putNetwSrvParamprimitive: (null)
#DEBUG# u_disc_req putNetwSrvParamstart(TU3901); to(discovering); }

 VTASK AFTER ACTIONS :
 vtask      : trans_1
 state      : discovering
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        1 geran_cov            :        1
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :        0
              paramMNC         :        0 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 timers     :                 length  left      inst
               TU3901          30.000s 29.999s   8a55fe0


**********************************************************************
```

```
 MESSAGE

 source     : trans_1
 destination: netw
 primitive  : t_dtr     to receiver's RES1 interface
 src cepid  : -
 dest cepid : -
 time       : 6.1.2006 16.39:18.823.539
 parameters :

#DEBUG# printNetwSrv
 Source address:
 Destination address:
 Framedata:
 (printing 52/52 bytes)
 00 32 01 01 01 06 89 34 47 76 32 84 02 01 00 06 ?2??????4Gv2?????
 01 00 04 02 03 4B 05 05 00 4A 44 00 04 29 01 00 ?????K???JD??)??
 15 01 00 0B 01 00 09 05 21 AF 73 F1 4C 61 05 21 ????????!?s?La?!
 7B 2D 7D 23                                     {-}#


 VTASK BEFORE RECEIVING THE MESSAGE:
 vtask      : netw
 state      : connected
 variables  : destinationAddr      :   (null) sourceAddr           :
(null)

              frame                :      0x0 ,sz 0
 timers     :

#DEBUG# getNetwSrv
 VTASK AFTER GETPARAMETERS :
 vtask      : netw
 state      : connected
 variables  : destinationAddr      :   (null) sourceAddr           :
(null)

              frame            :0x8a561c0 ,sz 52   00 32 01 01 01 06...
 state      : connected
 input      : RES1.t_dtr
 actions    :

                { woutput("
#DEBUG# got RES1.t_dtr");
#DEBUG# got RES1.t_dtr
                  RES2.t_dti;
#DEBUG# putNetwSrv}

 VTASK AFTER ACTIONS :
 vtask      : netw
 state      : connected
 variables  : destinationAddr      :   (null) sourceAddr           :
(null)

              frame                :      0x0 ,sz 0
 timers     :


**********************************************************************
```

```
 MESSAGE

 source     : netw
 destination: trans_2
 primitive  : t_dti     to receiver's DOWN interface
 src cepid  : -
 dest cepid : -
 time       : 6.1.2006 16.39:18.825.310
 parameters :

#DEBUG# printNetwSrv
 Source address:
 Destination address:
 Framedata:
 (printing 52/52 bytes)
 00 32 01 01 01 06 89 34 47 76 32 84 02 01 00 06 ?2??????4Gv2?????
 01 00 04 02 03 4B 05 05 00 4A 44 00 04 29 01 00 ?????K???JD??)??
 15 01 00 0B 01 00 09 05 21 AF 73 F1 4C 61 05 21 ????????!?s?La?!
 7B 2D 7D 23                                     {-}#


 VTASK BEFORE RECEIVING THE MESSAGE:
 vtask      : trans_2
 state      : idle
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        0 geran_cov            :        0
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :      843
              paramMNC         :       74 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 timers     :

 VTASK AFTER GETPARAMETERS :
 vtask      : trans_2
 state      : idle
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        0 geran_cov            :        0
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :      843
              paramMNC         :       74 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :0x8a561c0 ,sz 52   00 32 01 01 01 06...
              userData         :      0x0 ,sz 0
 state      : idle
 input      : DOWN.t_dti
 actions    :

                { decode(netData);

DECODE!
URR PDU: msg_type 0x1 length 50
```

```
Framelength >0 (48)
IEI_counter=1
CHECK_IEI_AND_LEN
   Length from frame: 6
   IEI: 1
    Len:6

      DEBUG:Mobile Identity: length = 6
      IMSI
              Length = 11
              Number = 84374672348

Framelength >0 (40)
IEI_counter=2
CHECK_IEI_AND_LEN
   Length from frame: 1
   IEI: 2
    Len:1

UMA Release Indicator IE:
      lenght = 1
      Release 1

Framelength >0 (37)
IEI_counter=3
CHECK_IEI_AND_LEN
   Length from frame: 1
   IEI: 6
    Len:1
GSM Coverage Indicator
      GCI = Normal Service in GERAN

Framelength >0 (34)
IEI_counter=4
CHECK_IEI_AND_LEN
   Length from frame: 2
   IEI: 4
    Len:2
Cell Identity:
      CI = 843

Framelength >0 (30)
IEI_counter=5
CHECK_IEI_AND_LEN
   Length from frame: 5
   IEI: 5
    Len:5

Location Area Identification:
      MCC = 74
      MNC = 68
      LAC = 4

Framelength >0 (23)
IEI_counter=6
CHECK_IEI_AND_LEN
   Length from frame: 1
   IEI: 41
    Len:1
```

```
Framelength >0 (20)
IEI_counter=7
CHECK_IEI_AND_LEN
   Length from frame: 1
   IEI: 21
    Len:1
Register Reject Cause:
      Cause = Network congestion

Framelength >0 (17)
IEI_counter=8
CHECK_IEI_AND_LEN
   Length from frame: 1
   IEI: 11
    Len:1
Redirection Counter:
      value = 0

Framelength >0 (14)
IEI_counter=9

DEC IP!!
      UNC SGW IP
      DEBUG: Decoded IPaddress: 76.241.115.175

Framelength >0 (7)
IEI_counter=10

DEC IP!!
      UNC IP
      DEBUG: Decoded IPaddress: 35.125.45.123

Returning pdu
to(idle); }

 state      : idle
 input      : PEER.DISC_REQ
 actions    :

                { initUNC();
      DEBUG:GLOBAL UNC IP: 595406203, var->unc_ip : 595406203
      DEBUG: Congestion: 0, IMSIBLOCK 0
                  if(acceptMSdis())
                  { woutput("
DISCOVERY REQUEST ACCEPTED!");
DISCOVERY REQUEST ACCEPTED!
                    encode(DISC_ACP);
###UNC IP ENC !
      DEBUG: encIP
###UNC SGW IP ENC !
      DEBUG: encIP}
                  DOWN.t_dtr;
#DEBUG# beginning at putNetwSrvParamprimitive: (null)
#DEBUG# u_disc_req putNetwSrvParamto(ms_disc); }

 VTASK AFTER ACTIONS :
 vtask      : trans_2
 state      : ms_disc
 variables  : congestion       :        0 debug1              :        1
              debug2           :        1 debug3              :        1
```

```
              disc_rejected    :        0 geran_cov            :        0
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :      843
              paramMNC         :       74 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 timers     :


**************************************************************************

 MESSAGE

 source     : trans_2
 destination: netw
 primitive  : t_dtr     to receiver's RES2 interface
 src cepid  : -
 dest cepid : -
 time       : 6.1.2006 16.39:18.831.086
 parameters :

#DEBUG# printNetwSrv
 Source address:
 Destination address:
 Framedata:
 (printing 18/18 bytes)
 00 10 01 02 09 05 21 AF 73 F1 4C 61 05 21 7B 2D ??????!?s?La?!{-
 7D 23                                           }#


 VTASK BEFORE RECEIVING THE MESSAGE:
 vtask      : netw
 state      : connected
 variables  : destinationAddr      :   (null) sourceAddr           :
(null)

              frame                :      0x0 ,sz 0
 timers     :

#DEBUG# getNetwSrv
 VTASK AFTER GETPARAMETERS :
 vtask      : netw
 state      : connected
 variables  : destinationAddr      :   (null) sourceAddr           :
(null)

              frame            :0x8a561c0 ,sz 18   00 10 01 02 09 05...
 state      : connected
 input      : RES2.t_dtr
 actions    :

                { woutput("
#DEBUG# got RES2.t_dtr");
#DEBUG# got RES2.t_dtr
                  RES1.t_dti;
#DEBUG# putNetwSrv}
```

```
 VTASK AFTER ACTIONS :
 vtask      : netw
 state      : connected
 variables  : destinationAddr      :   (null) sourceAddr          :
(null)

              frame               :      0x0 ,sz 0
 timers     :


**************************************************************************

 MESSAGE

 source     : netw
 destination: trans_1
 primitive  : t_dti     to receiver's DOWN interface
 src cepid  : -
 dest cepid : -
 time       : 6.1.2006 16.39:18.832.734
 parameters :

#DEBUG# printNetwSrv
 Source address:
 Destination address:
 Framedata:
 (printing 18/18 bytes)
 00 10 01 02 09 05 21 AF 73 F1 4C 61 05 21 7B 2D ??????!?s?La?!{-
 7D 23                                           }#


 VTASK BEFORE RECEIVING THE MESSAGE:
 vtask      : trans_1
 state      : discovering
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        1 geran_cov            :        1
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :        0
              paramMNC         :        0 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 timers     :                length  left      inst
               TU3901         30.000s 29.989s   8a55fe0

 VTASK AFTER GETPARAMETERS :
 vtask      : trans_1
 state      : discovering
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        1 geran_cov            :        1
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :        0
              paramMNC         :        0 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
```

```
              netData          :0x8a561c0 ,sz 18   00 10 01 02 09 05...
              userData         :      0x0 ,sz 0
 state      : discovering
 input      : DOWN.t_dti
 actions    :

                { decode(netData);

DECODE!
URR PDU: msg_type 0x2 length 16

Framelength >0 (14)
IEI_counter=1

DEC IP!!
      UNC SGW IP
      DEBUG: Decoded IPaddress: 76.241.115.175

Framelength >0 (7)
IEI_counter=2

DEC IP!!
      UNC IP
      DEBUG: Decoded IPaddress: 35.125.45.123

Returning pdu
stop(TU3901); to(discovering); }

 state      : discovering
 input      : PEER.DISC_ACP
 actions    :

                { woutput("

MS GOT DISCOVERY ACCEPT FROM UNC!");

MS GOT DISCOVERY ACCEPT FROM UNC!
                  to(discovering); }

 VTASK AFTER ACTIONS :
 vtask      : trans_1
 state      : discovering
 variables  : congestion       :        0 debug1               :        1
              debug2           :        1 debug3               :        1
              disc_rejected    :        1 geran_cov            :        1
              gsm_rr_state     :        0 imsi_block           :        0
              msg_type         :        0 paramMCC             :        0
              paramMNC         :        0 prot_disc_and_skip_i :        0
              redir_counter    :        0 reg_rejected         :        0
              rel_ind          :        0 rr_cause             :        0
              utran_cov        :        0
              netData          :      0x0 ,sz 0
              userData         :      0x0 ,sz 0
 timers     :

: macro end

: E

 wexit() called ... exiting now!
```